PCS-AC: COMPILING OUTPUT-SENSITIVE ACTIONS TO CORRECT MEDICAL PREDICTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Correcting a frozen model for medical decision-making requires interventions that fix errors without disrupting predictions that are already correct. A shared representation edit is often insufficient because the same direction may help some inputs but harm others. We introduce Predictable Corrective Subspaces via Action Compilation (PCS-AC), an output-sensitive framework that learns input-dependent corrective interventions in hidden representations. Specifically, a budget-weighted Jacobian identifies an output-sensitive subspace, within which source labels define bounded corrective actions. We compile these actions into an input-dependent low-rank controller via reduced-rank regression and refine it through the frozen backbone to improve predictions while preserving source margins. At inference, the controller uses only the current hidden state, without labels, gradients, or an additional backbone pass.

On 2,100 retrospective radiologic questions, PCS-AC reaches 80.95% and 81.62% accuracy with Qwen2.5-VL-7B and InternVL3.5-8B, improving Native by 12.19 and 42.67 percentage points. Under matched source supervision, response subspace, and refinement budget, it outperforms a learned shared direction by 13.31 and 28.45 points. Matched-capacity comparisons further favor output-sensitive construction over random and activation-PCA pipelines. An independently fitted text controller reaches 95.03% on 3,360 controlled synthetic assertions, a 5.98-point improvement over Native. These results show that compiling output-sensitive corrective actions into conditional controllers enables more selective correction of frozen models.

1 INTRODUCTION

For medical tasks, model adaptation must improve predictive performance without introducing unnecessary errors into decisions that are already correct, since newly induced mistakes can be consequential. This makes correction more demanding than simply shifting model behavior. Representation intervention (Wu et al., 2024) offers a lightweight solution by modifying selected hidden states while keeping the backbone frozen. Existing methods provide low-rank and input-dependent mechanisms for such interventions (Lee et al., 2025; Wang et al., 2025a; Do et al., 2025), but a central question remains for correction: *how should supervision be converted into an output-relevant, input-specific hidden-state action for a particular example?*

This question involves two coupled challenges. First, choosing a compact intervention space does not determine which directions influence the model's answer margin for a given input. Representation variance alone does not characterize this output sensitivity: directions along which hidden states vary substantially need not be those along which small perturbations most strongly change the answer margin. Second, even within a decision-relevant space, the appropriate movement may vary across examples in both direction and magnitude, making a single shared steering direction insufficient in general. Effective correction therefore requires identifying output-sensitive directions, determining example-specific movements within them from source supervision, and learning to predict those movements from the representation itself.

To address these challenges, we introduce **P**redictable **C**orrective **S**ubspaces via **A**ction **C**ompilation (PCS-AC), an output-sensitive framework for learning input-specific corrections to frozen models. Our central idea is to *construct corrective actions before learning to predict them*. Specifically, on la-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

beled source examples, the model's answer-margin response identifies representation directions that can influence the decision, while supervision determines the desired movement within this decision-relevant space. These source-local actions are compiled into a compact conditional controller that predicts an intervention from the current hidden state. Because the constructed actions rely on a local linearization, we refine the compiled controller through the frozen backbone to account for its nonlinear response. A source-margin regularizer discourages weakening initially correct source decisions during refinement. Once fitted, the controller maps the hidden state to an intervention, requiring no labels, gradients, or an additional backbone pass at inference.

Across three independently fitted backbones, PCS-AC consistently improves prediction accuracy. On a 2,100-question RadImageNet-VQA project holdout (Butsanets et al., 2026), accuracy rises from 68.76% to 80.95% for Qwen2.5-VL-7B (Bai et al., 2025) and from 38.95% to 81.62% for InternVL3.5-8B (Wang et al., 2025b). A separately fitted text controller improves LLaMA-3.1-8B (Grattafiori et al., 2024) from 89.05% to 95.03% on 3,360 controlled synthetic assertions. For Qwen and InternVL, respectively, matched controls test the two design choices motivated above: input-specific conditional actions outperform a learned shared direction by 13.31 and 28.45 points, while output-sensitive construction exceeds activation-PCA pipelines by 8.17 and 35.98 points. The VLM controllers add under one millisecond of median forward latency for batch-one binary scoring.

In summary, our contributions are threefold:

- **Corrective-action compilation.** We formulate frozen-model correction as predicting source-local actions from hidden states, converting label- and gradient-informed targets into deployable interventions.
- **Output-sensitive conditional intervention.** PCS-AC predicts bounded, input-dependent corrections in output-relevant subspaces and penalizes weakening initially correct source margins during refinement.
- **Performance and mechanism evidence.** Experiments across two VLMs and one text model show accuracy gains, while matched controls and ablations support the value of conditional action prediction and output-sensitive action construction.

2 Related work

Representation intervention and conditional steering. LoRA learns low-rank weight updates (Hu et al., 2022), whereas ReFT learns interventions on hidden representations (Wu et al., 2024). Activation steering methods modify representations using contrastive activation directions (Rimsky et al., 2024) or attention-head interventions for truthfulness (Li et al., 2023b). Input-dependent approaches gate behavior vectors (CAST; Lee et al. 2025), scale activation coordinates (SADI; Wang et al. 2025a), or retrieve steering vectors from episodic memory (DSEM; Do et al. 2025). AxBench highlights the strength of simple steering baselines (Wu et al., 2025). PCS-AC uses source labels and answer-margin sensitivity to construct corrective targets for a conditional controller.

Output-guided correction and amortized control. PPLM steers generation with attribute gradients (Dathathri et al., 2020), while active subspaces identify output-sensitive directions from gradient second moments (Constantine et al., 2014). Policy distillation amortizes teacher behavior into a learned policy (Rusu et al., 2016). More recently, L2S predicts steering vectors from contrastive activation targets (Parekh et al., 2025), and CASAL amortizes activation steering through training (Yang et al., 2026). PCS-AC constructs bounded source-local actions from budget-weighted margin derivatives and compiles them by reduced-rank regression. The controller is refined through the frozen backbone; inference uses hidden states alone.

Model editing and medical adaptation. Model editing methods modify model behavior while seeking to preserve unrelated predictions. MEND transforms fine-tuning gradients into localized weight edits (Mitchell et al., 2022). In medicine, models such as LLaVA-Med (Li et al., 2023a) and Med-Flamingo (Moor et al., 2023) adapt general-purpose backbones through biomedical instruction tuning or continued pretraining. PCS-AC instead learns compact representation-level corrections with frozen backbone weights and penalizes weakening initially correct source margins during refinement.

3 BINARY CORRECTION OF FROZEN MODELS

Let z be an input, a_+, a_- its candidate answers, and $y \in \{-1, +1\}$ its label. The native margin and prediction are

$$m^0(z) = \log p(a_+ \mid z) - \log p(a_- \mid z), \qquad \widehat{y}^0 = \begin{cases} +1 & m^0 \geq 0, \\ -1 & m^0 < 0. \end{cases} \tag{1}$$

For visual questions, $z = (I, q)$ and the candidates are yes/no; for text, z contains a statement and target entity with Present/Absent candidates. All candidates are single tokens, so the margin equals the difference of their next-token logits. We evaluate restricted binary decisions, not unrestricted answer generation.

At one decoder layer, let $h \in \mathbb{R}^d$ be the last prompt-token state. We replace it by $h + \delta$ and run the remaining frozen network to obtain $m(z; \delta)$. For a learned action $\delta_\theta(h)$, write $m^\theta = m(z; \delta_\theta(h))$. A *repair* changes a Native error to a correct answer; *damage* reverses a Native-correct answer. Their difference is *net correction*, so accuracy changes by $(\text{repair} - \text{damage})/n$ on n examples. All reported decisions use actual intervened-model outputs.

4 OUTPUT-SENSITIVE ACTION COMPILATION

PCS-AC separates source-local action construction from conditional action prediction (Figure 1). Output response defines the intervention space; source supervision defines bounded actions within it. Reduced-rank regression compiles these actions into a hidden-state controller, which is then refined through the frozen model.

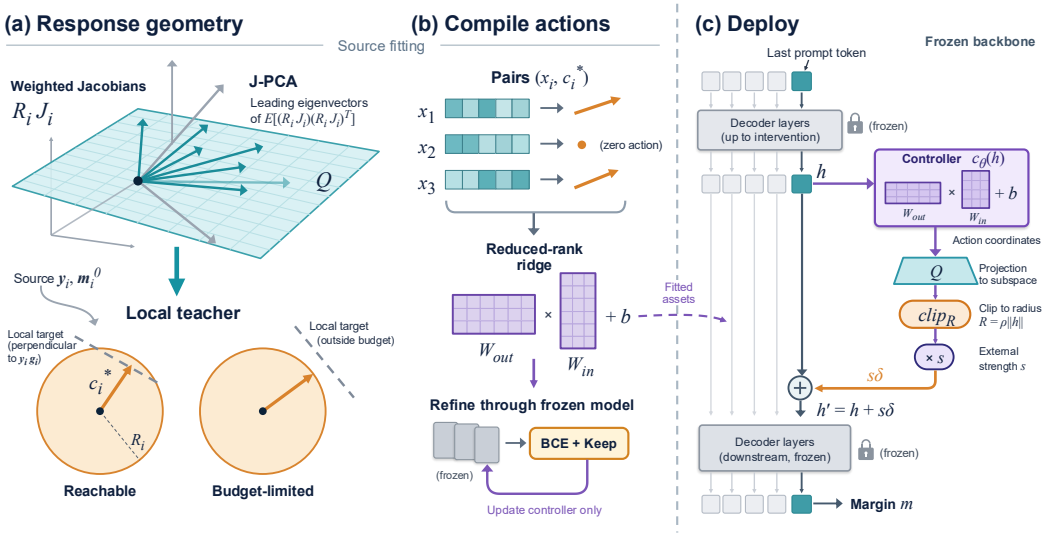


Figure 1: Output-sensitive action compilation. (a) Source labels and margin gradients define bounded actions in a response subspace. (b) Reduced-rank regression compiles the actions into a hidden-state controller, then refinement optimizes its actual model predictions. (c) Deployment projects the predicted coordinates through Q, clips the displacement, applies external strength s, and edits the last prompt-token state within one backbone forward pass. Assets are fitted independently for each backbone.

4.1 OUTPUT-SENSITIVE RESPONSE SUBSPACE

We select directions by their influence on the answer margin rather than representation variance. Let $\mathcal{D}_s = \{(z_i, y_i)\}_{i=1}^N$ be the labeled source data. For each example, we cache the native hidden state

h_i, margin m_i^0, and margin gradient $J_i = \nabla_{h_i} m_i^0$. The local response to an intervention is

$$m(z_i; \delta) \approx m_i^0 + J_i^\top \delta. \tag{2}$$

We impose a sample-relative movement budget $R_i = \rho \|h_i\|_2$, with $\rho = 0.1$, and restrict actions to an orthonormal basis $Q \in \mathbb{R}^{d \times k}$. For a fixed Q, the largest absolute first-order margin change within this budget is $R_i \left\|Q^\top J_i\right\|_2$. We therefore select Q to retain the average squared response capacity:

$$Q \in \arg\max_{Q^\top Q = I_k} \frac{1}{N} \sum_{i=1}^{N} R_i^2 \left\|Q^\top J_i\right\|_2^2, \qquad M = \frac{1}{N} \sum_{i=1}^{N} R_i^2 J_i J_i^\top. \tag{3}$$

The top $k = 64$ eigenvectors of M solve this objective. We call this construction budget-weighted J-PCA. The moment is uncentered: a response direction shared across examples should remain available for intervention. Weighting by R_i^2 accounts for the movement each example permits. Unlike activation PCA, the criterion measures sensitivity of the answer margin rather than variance of the representation. Labels determine the correction targets below, but do not enter Eq. 3.

4.2 Bounded corrective-action targets

Within the response space, source labels and local margin responses specify both a corrective direction and a required movement. Write $g_i = Q^\top J_i$ and let $\Delta_i = [\gamma - y_i m_i^0]_+$ be the gap to a desired signed margin $\gamma > 0$. We set γ to the median absolute native margin among correctly classified source examples, floored at 10^{-6}. For a nonzero response, the teacher action is

$$c_i^* = y_i \min\left\{ \frac{\Delta_i}{\|g_i\|_2^2}, \frac{R_i}{\|g_i\|_2} \right\} g_i, \qquad \delta_i^* = Q c_i^*. \tag{4}$$

If $\Delta_i \le R_i \|g_i\|_2$, this is the minimum-norm action satisfying $y_i g_i^\top c \ge \Delta_i$. Otherwise, it exhausts the budget in the direction of maximum signed local response. An example already exceeding γ receives a zero target; a correct but low-margin example may still receive a reinforcing action. We set $c_i^* = 0$ when $\|g_i\|_2 \le 10^{-12}$. Appendix B gives the derivation.

These actions serve as vector-valued supervision for the controller below. They depend on source labels and local derivatives, and their optimality holds for the linearized margin. Compilation predicts them from h; refinement accounts for the nonlinear model response.

4.3 Low-rank conditional compilation

We turn source-local actions into deployable predictions by fitting their coordinates from normalized hidden states. Let $x(h) = (u(h) - \mu)/\sigma$, where $u(h)$ is RMS-normalized h, μ is its source mean, and σ is a single source-fitted scale (Appendix C). We fit the teacher coordinates by reduced-rank ridge regression:

$$\min_{W, b:\, \mathrm{rank}(W) \le r} \frac{1}{N} \sum_{i=1}^{N} \|c_i^* - W x(h_i) - b\|_2^2 + \eta \|W\|_F^2, \qquad r = 8, \quad \eta = 0.01. \tag{5}$$

The bias is unpenalized. We solve this auxiliary objective in closed form and factor its solution as $W = W_{\mathrm{out}} W_{\mathrm{in}}$, where $W_{\mathrm{in}} \in \mathbb{R}^{r \times d}$ and $W_{\mathrm{out}} \in \mathbb{R}^{k \times r}$. Compilation thus initializes a predictor of vector-valued actions, including examples whose required action is zero.

The controller predicts coordinates and clips the resulting hidden-state displacement:

$$c_\theta(h) = W_{\mathrm{out}} W_{\mathrm{in}} x(h) + b, \tag{6}$$

$$\delta_\theta(h) = \mathrm{clip}_{\rho\|h\|_2}\big(Q c_\theta(h)\big), \quad \mathrm{clip}_R(v) = v \min\left\{1, \frac{R}{\max(\|v\|_2, 10^{-12})}\right\}. \tag{7}$$

The two factors have no intervening nonlinearity. The predicted direction and raw magnitude vary with the input; clipping enforces the movement budget even when the regression extrapolates. Throughout refinement, Q, μ and σ remain fixed. The representation is low-dimensional but dense: no per-example coordinate sparsity is imposed.

Compilation provides an executable controller, but minimizing action-prediction error does not directly optimize the intervened model's predictions. We therefore use this solution to initialize refinement through the frozen backbone.

4.4 REFINEMENT AND DEPLOYMENT

We refine the compiled controller through the frozen backbone to optimize its actual nonlinear predictions. Let $m_i^\theta = m(z_i; \delta_\theta(h_i))$ be the actual intervened model margin, and let $\mathcal{K}_0 = \{i : \widehat{y}_i^0 = y_i\}$ be the fixed Native-correct source set. Starting from the compiled controller, we minimize

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathrm{softplus}(-y_i m_i^\theta) + \frac{\lambda}{|\mathcal{K}_0|}\sum_{i\in\mathcal{K}_0} \frac{[y_i(m_i^0 - m_i^\theta)]_+}{|m_i^0| + \gamma}. \tag{8}$$

The prediction term trains correction through the frozen downstream network. The second term, *Keep*, penalizes erosion of Native-correct signed margins, normalized by $|m_i^0| + \gamma$; improved margins incur no penalty. We use $\lambda = 1$ (NoKeep: zero) and the fixed population factor $N/|\mathcal{K}_0|$ in minibatches. This source regularizer does not enforce target preservation; its paired effects are evaluated in Appendix K.

Full-controller and output-head refinement. VLMs update $(W_{\mathrm{in}}, W_{\mathrm{out}}, b)$; text freezes the compiled input map and updates only (W_{out}, b). The respective refinement parameter counts are $r(d+k)+k$ and $kr + k = 576$ for $k = 64, r = 8$. Both retain the source-fitted basis and feature statistics, and text also stores its frozen input map. Appendix A specifies model-dependent schedules and full asset counts.

Deployment and strength selection. At inference, the current hidden state determines the action, and external strength $s \in [0, 1]$ scales it *after clipping*:

$$h' = h + s\ \mathrm{clip}_{\rho\|h\|_2}[Q\,(W_{\mathrm{out}} W_{\mathrm{in}} x(h) + b)]\,. \tag{9}$$

Thus $\|h' - h\|_2 \le s\rho\,\|h\|_2$ and $s = 0$ recovers Native. Main VLMs use $s = 1$; text uses development-selected $s = .75$. The controller learns input-dependent direction and raw amplitude, whereas s is selected externally. Deployment uses no labels, Jacobians, gradient updates, or additional backbone pass. Controllers are fitted independently; labeled target-support adaptation is a separate setting (Appendix H).

Algorithm 1 summarizes how source supervision becomes a hidden-state intervention executed within one model forward pass.

Algorithm 1 PCS-AC: source fitting and deployment

Inputs: frozen backbone, labeled source $\mathcal{D}_s$, development data, layer, $k = 64$, $r = 8$, $\rho = .1$.

1. Cache each source example's native state h_i, margin m_i^0, and gradient J_i.
2. Set $R_i = \rho\,\|h_i\|_2$; fit Q from Eq. 3 and source statistics (μ, σ, γ).
3. Construct the bounded teacher coordinates c_i^* with Eq. 4.
4. Solve Eq. 5 and factor the result into $(W_{\mathrm{in}}, W_{\mathrm{out}}, b)$.
5. Fix the Native-correct source set $\mathcal{K}_0$; refine through the frozen model using Eq. 8. Update all controller factors for VLMs, or only (W_{out}, b) for text.
6. Freeze the fitted assets and deployment strength: $s = 1$ for the main VLMs, development-selected $s = .75$ for text.
7. For a new input, compute h at the chosen layer, apply Eq. 9 once, and continue the same forward pass to obtain the candidate margin.

5 EXPERIMENTS

We evaluate PCS-AC on visual and text-only prediction, test the contributions of conditional actions and output-sensitive construction, and measure inference overhead.

5.1 EXPERIMENTAL SETUP

Data and models. We use the binary abnormality-detection task from RadImageNet-VQA (Butsanets et al., 2026). Our splits contain 3,841 source question-answer pairs, 647 source-development

pairs (C), and a 2,100-question evaluation across seven anatomical environments. These splits come from publisher training shards. The text task uses controlled synthetic Present/Absent assertions created within this project: 512 balanced source examples, 600 development examples, and 3,360 evaluation examples in four previously studied subsets. All evaluations are retrospective. Source, development, and evaluation sets have disjoint sample IDs, known dependency groups, and exact image or text–entity hashes. Appendix D summarizes all cohorts, including the original VQA-RAD binary test subset (Lau et al., 2018) used for transfer.

Both tasks use fixed candidate tokens: visual `yes`/`no` and text `Present`/`Absent` for a named entity. For example, "The evidence can indicate effusion while leaving fracture unsupported" has label `Absent` for `fracture` (Appendix O).

We fit separate controllers to Qwen2.5-VL-7B-Instruct (Bai et al., 2025), InternVL3.5-8B-HF (Wang et al., 2025b), and LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024) at zero-based decoder layers 14, 18, and 2. All use a 64-dimensional response basis, rank eight, $\rho = .1$, $\lambda = 1$, batch size eight, seeds 17/29, and the final checkpoint. VLMs refine all controller factors for two epochs; text refines the controller's 576-parameter output head for eight epochs. Deployment uses $s = 1$ for VLMs and development-selected $s = .75$ for text. Backbones remain frozen in FP32. Appendices A and E give hyperparameters, prompts, and preprocessing.

Comparisons and metrics. We compare PCS-AC with Native, LoReFT, fixed-direction steering, and source-only calibration. Native is the unmodified model; rank-four LoReFT (Wu et al., 2024) is a learned hidden-state intervention. Text mean-difference steering, a learned shared direction, and a threshold selected on C test whether fixed actions or output calibration explain the gains. AxBench also evaluates mean-difference steering and learned representation interventions (Wu et al., 2025). Source-fitted interventions use the same labeled data. All main-table methods use identical candidate scoring and a zero decision threshold. Interventions share the layer, prompt-token position, and frozen downstream model within each backbone. LoReFT uses the same prediction-plus-Keep objective and norm cap, two epochs, and learning rate .002. VLM update budgets match; text uses eight PCS-AC epochs versus two LoReFT epochs. Appendix J gives the development-set selection procedure.

Metrics. Binary accuracy measures the fraction of correct answers. Repair and damage counts compare each intervention with Native, separating corrected errors from new errors. Their difference divided by cohort size gives the accuracy change. Balanced accuracy averages the two class recalls to account for label imbalance. Paired timings quantify inference overhead. We report two-seed means for learned methods; accuracy differences use unrounded scores.

Ablations. We test five design choices: shared versus conditional actions; random, activation-PCA, and J-PCA spaces; compilation and initialization; Keep versus NoKeep; and deployment strength. Shared-direction controls match source supervision, response space, and refinement schedule. Space controls match controller capacity and refinement budget, with teacher targets and regression initialization recomputed for each space. Tables 3–4 report the main comparisons. Appendices J–L give matched text ablations, preservation results, and comparisons on a common strength grid.

Evaluation across models and tasks. We test visual correction on two backbones, fitting a separate controller to each, and report results across seven anatomical environments. The independently fitted text controller tests the method on a second input modality. We report environment and text-subset accuracies alongside pooled scores to show where the overall gains occur. Development selection uses data separate from the evaluation cohorts.

5.2 Visual correction across backbones

PCS-AC improves pooled visual accuracy on both backbones (Table 1), reaching **80.95% on Qwen and 81.62% on InternVL**. The gains are **12.19 and 42.67 percentage points** over Native and 11.19 and 7.74 points over LoReFT. Relative to Native, these gains correspond to 256 and 896 net corrected decisions. PCS-AC improves on Native in six of seven environments for Qwen and all seven for InternVL; it exceeds LoReFT in five and six environments, respectively. On Qwen, MRI abdomen accuracy rises from 25.00% to 65.00%, and MRI brain accuracy from 48.00% to 80.50%.

Table 1: Visual correction across two frozen backbones. Accuracy (%) on seven RadImageNet-VQA environments and their pooled 2,100-question evaluation. Controllers are fitted independently; bold marks the best listed accuracy per backbone and environment.

		Qwen2.5-VL-7B			InternVL3.5-8B		
Environment	n	Native	LoReFT	PCS-AC (Ours)	Native	LoReFT	PCS-AC (Ours)
CT abdomen	300	52.67	53.50	**59.00**	52.00	60.00	**63.83**
MRI abdomen	300	25.00	25.17	**65.00**	80.33	80.83	**82.17**
MRI ankle/foot	300	85.67	87.50	**90.00**	20.33	70.33	**81.50**
MRI brain	300	48.00	48.83	**80.50**	65.67	78.17	**86.33**
MRI knee	300	90.67	92.83	**93.33**	17.00	84.00	**87.17**
MRI shoulder	300	**95.00**	**95.00**	94.33	22.33	**92.33**	89.50
MRI spine	300	84.33	**85.50**	84.50	15.00	51.50	**80.83**
Overall	2,100	68.76	69.76	**80.95**	38.95	73.88	**81.62**

Overall pools predictions across the seven environments in the 2,100-QA holdout. Learned methods report two-seed means and use two epochs, $s = 1$, the same scoring interface, and a zero decision threshold.

5.3 TEXT-ONLY ASSERTION CORRECTION

Table 2: Text-only assertion correction with LLaMA-3.1-8B-Instruct. Accuracy (%) on four controlled synthetic subsets and their pooled 3,360-example evaluation; Δ is PCS-AC minus Native (pp). Bold marks the best listed accuracy per row.

Subset	n	Native	Mean-diff.	LoReFT	PCS-AC (Ours)	Δ
V2	840	93.93	93.93	**99.23**	99.05	+5.12
V3	840	88.21	88.21	**90.60**	90.48	+2.26
V5	840	88.10	87.98	92.50	**95.77**	+7.68
V6	840	85.95	85.95	93.45	**94.82**	+8.87
Overall	3,360	89.05	89.02	93.94	**95.03**	+5.98

V2/V3/V5/V6 come from the synthetic collection used in earlier experiments. Mean-diff. denotes source mean-difference steering. Learned methods report two-seed means. PCS-AC uses the development-selected eight-epoch output-head configuration ($s = .75$); fixed LoReFT uses two epochs ($s = 1$), so optimization budgets differ. Component variants appear in Table 4(b).

PCS-AC also improves text-only predictions. An independently fitted LLaMA controller achieves **95.03%** on 3,360 controlled synthetic assertions, exceeding Native by **5.98 points** and LoReFT by 1.09 points (Table 2). Accuracy improves over Native in all four subsets, yielding 201 net corrections. Mean-difference steering remains close to Native at 89.02%. During refinement, the text controller updates only its 576-parameter output head; the compiled input map remains fixed. Appendix O illustrates individual visual and text corrections.

5.4 CONDITIONAL ACTIONS IMPROVE NET CORRECTION

Input-dependent actions outperform the learned shared direction by **13.31 points on Qwen and 28.45 on InternVL**, with the same source supervision, J-PCA64 space, refinement schedule, and strength (Table 3(a)). Net corrections rise from -23.5 to 256 on Qwen and from 298.5 to 896 on InternVL. When C selects strength separately for each method, the gains remain 12.929 and 28.452 points (Appendix F).

PCS-AC also exceeds Native after source-only threshold calibration. Selecting the threshold on C raises Native accuracy to 69.000% on Qwen and 70.381% on InternVL. PCS-AC exceeds these calibrated baselines by **11.952 and 11.238 points**, respectively (Table 3(b)). Changing the decision threshold alone does not account for PCS-AC's gains.

Table 3: Conditional actions and source-only calibration controls on the same 2,100-question hold-out. Panel (a) matches source supervision, J-PCA64 space, refinement schedule, Keep, and $s = 1$. Panel (b) changes only Native's threshold, selected on source-development C (647 QA). Accuracy is in percent; repair, damage, and net correction are two-seed mean counts.

(a) Conditional prediction versus a learned shared direction

Model	Controller	Accuracy	Repair	Damage	Net
Qwen	Shared direction	67.643	54.0	77.5	−23.5
Qwen	PCS-AC (Ours)	**80.952**	351.5	95.5	256.0
InternVL	Shared direction	53.167	340.0	41.5	298.5
InternVL	PCS-AC (Ours)	**81.619**	960.0	64.0	896.0

(b) Source-only threshold calibration

Model	Native	Native + C threshold	PCS-AC (Ours)	Δ
Qwen	68.762	69.000	**80.952**	+11.952
InternVL	38.952	70.381	**81.619**	+11.238

Net is repair minus damage; panel (b)'s Δ is PCS-AC minus calibrated Native (pp). Bold marks the higher accuracy per model. Shared directions train 64 parameters versus 29,248/33,344 for PCS-AC on Qwen/InternVL; initializations also differ. PCS-AC uses $s = 1$ and a zero threshold. Appendix F gives training and source-C strength-selection details.

5.5 Output-sensitive construction and compilation

At matched controller capacity and refinement budget, controllers built with J-PCA exceed those built with activation PCA by **8.17 and 35.98 points** and random subspaces by **9.52 and 39.86 points** on Qwen and InternVL, respectively (Table 4(a)). We recompute the teacher targets and regression initialization for each space. The comparison includes both the choice of subspace and the action targets constructed within it. On InternVL, J-PCA exceeds both alternatives in all seven environments (Appendix G).

Table 4: Output-sensitive construction and compilation. Mean accuracy (%). Panel (a) matches dimension, controller capacity, and refinement budget; each space refits its teacher and compiler. Panel (b) separates compilation from refinement and compares output initializations.

(a) Choice of action space

Action space	Qwen	InternVL
Random orthogonal	71.43	41.76
Activation PCA	72.79	45.64
J-PCA (Ours)	**80.95**	**81.62**

(b) Compilation and initialization

Model	Compiled only	Zero-output + refine	PCS-AC (Ours)
Qwen	80.38	79.17	**80.95**
InternVL	65.76	80.55	**81.62**
Llama	93.12	**95.86**	95.03

Panel (a): dimension 64, rank 8, two epochs, Keep, and $s = 1$; one random basis per backbone shared across training seeds. Panel (b): VLMs use $s = 1$; text refinements use $s = .75$, compiled-only $s = 1$, each development-selected. The VLM zero-output control has a random trainable input map; text retains its frozen compiled input map. Bold marks the best listed accuracy per backbone and panel. Appendices G and I give full protocols.

From compiled actions to refined predictions. Before refinement, the compiled controllers reach 80.38%, 65.76%, and 93.12% on Qwen, InternVL, and LLaMA, respectively (Table 4(b)). All three improve on Native before further training through the backbone. Much of Qwen's gain is present at this stage; refinement through the frozen model adds another 15.86 points on InternVL.

After refinement, compiled initialization exceeds the zero-output controls by 1.79 points on Qwen and 1.07 on InternVL. These VLM controls use random input maps. For text, zero-initializing the output head while retaining the frozen compiled input map reaches 95.86%. The compiled input map is unchanged in this text ablation (Appendix I).

Paired correction and class balance. On InternVL, PCS-AC repairs 960 Native errors and changes 64 correct answers to errors. The activation-PCA controller makes 147.5 repairs and introduces seven errors, giving PCS-AC 755.5 additional net corrections. Balanced accuracy also favors PCS-AC: 83.69% versus 59.86% for activation PCA and 77.89% for fixed LoReFT. Compared with LoReFT, PCS-AC repairs 162 more errors and introduces 0.5 fewer errors (Appendix G).

5.6 Lightweight deployment after compilation

PCS-AC computes its intervention within the backbone's original forward pass, without inference-time labels or Jacobians. For batch-one FP32 binary-scoring prefill, the median paired increase in forward latency is **0.78 ms on Qwen, 0.50 ms on InternVL, and 5.18 ms on LLaMA** (Table 5). Median relative overheads are 0.36%, 0.07%, and 4.04%, respectively. Both VLMs gain accuracy with less than one millisecond of median additional forward latency. Appendix M gives hardware, end-to-end timings, and the full comparison. These timings exclude controller fitting.

Table 5: Measured deployment overhead of PCS-AC. Medians of paired extra forward latency and relative overhead against Native under batch-one FP32 binary-decision prefill. Each backbone uses 96 development inputs and three repetitions.

Backbone	Extra latency (ms)	Relative overhead (%)
Qwen2.5-VL-7B	+0.78	0.36
InternVL3.5-8B	+0.50	0.07
LLaMA-3.1-8B	+5.18	4.04

Differences and ratios are computed per input and repetition before taking medians. Full timings and the fixed LoReFT reference appear in Appendix M.

Stored parameters and timing. At deployment, we store the basis, both low-rank factors, the bias, and normalization statistics; matrix products cost $O(dr + rk + dk)$ per input. The text controller's 576 updated parameters exclude the stored frozen input map and basis. We time Native and PCS-AC on identical inputs and hardware, compute the latency difference for each input and repetition, then take the median. Source Jacobian extraction and controller refinement occur offline and are excluded from these timings. At inference, the fitted controller predicts actions directly from hidden states.

6 Discussion and conclusion

PCS-AC learns to predict corrective actions constructed from source labels and answer-margin gradients. The same procedure improves pooled accuracy on two vision–language backbones and a separately fitted text model. Shared-direction and action-space comparisons support the use of conditional actions and output-sensitive construction. Once fitted, the controller uses the current hidden state and adds little forward-pass overhead in the measured setting.

The results concern retrospective binary prediction, including synthetic text; clinical utility and patient-level independence remain untested. On same-domain visual data, Keep reduces new errors but also prevents repairs, lowering accuracy. Text initialization has a different effect from VLM initialization, and frozen transfer loses accuracy on VQA-RAD (Appendices I, K–L, and H). These findings make target-domain validation important even when a controller improves source-domain performance.

REPRODUCIBILITY STATEMENT

We specify the model configurations and training hyperparameters in Appendix A, data splits and selection rules in Appendix D, and prompts and preprocessing in Appendix E. Appendix M describes the timing protocol.

AI USE STATEMENT

We used generative AI to assist with code implementation and debugging, data preparation and review, and manuscript polishing. The authors reviewed the AI-assisted work and take responsibility for the final text, claims, and code.

ETHICS STATEMENT

In our main benchmark evaluations, PCS-AC improves overall accuracy on medical prediction tasks. Its preservation objective discourages weakening initially correct source predictions. We aim to support more reliable medical AI for the benefit of patients, clinicians, and the wider public. Clinical use would require further validation and assessment of whether interventions introduce or amplify bias across patient groups. We follow the model and dataset access terms and have obtained permission to display the selected RadImageNet-VQA images.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, et al. Qwen2.5-VL technical report, 2025. URL https://arxiv.org/abs/2502.13923.

Léo Butsanets, Charles Corbière, Julien Khlaut, Pierre Manceron, and Corentin Dancette. RadImageNet-VQA: A large-scale CT and MRI dataset for radiologic visual question answering. In *Proceedings of The 9th International Conference on Medical Imaging with Deep Learning*, volume 315 of *Proceedings of Machine Learning Research*, pp. 3036–3068. PMLR, 2026. URL https://proceedings.mlr.press/v315/butsanets26a.html.

Paul G. Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: Applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014. doi: 10.1137/130916138. URL https://epubs.siam.org/doi/10.1137/130916138.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1912.02164.

Van Dai Do, Quan Hung Tran, Svetha Venkatesh, and Hung Le. Dynamic steering with episodic memory for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 13731–13749. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.706. URL https://aclanthology.org/2025.findings-acl.706/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783v3.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jason J. Lau, Soumya Gayen, Asma Ben Abacha, and Dina Demner-Fushman. A dataset of clinically generated visual questions and answers about radiology images. *Scientific Data*, 5:180251, 2018. doi: 10.1038/sdata.2018.251. URL https://www.nature.com/articles/sdata2018251.

Bruce W. Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e2dd53601de57c773343a7cdf09fae1c-Abstract-Conference.html.

Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. LLaVA-Med: Training a large language-and-vision assistant for biomedicine in one day. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28541–28564, 2023a. doi: 10.52202/075280-1240.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 41451–41530, 2023b. doi: 10.52202/075280-1797.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.11309.

Michael Moor, Qian Huang, Shirley Wu, Michihiro Yasunaga, Yash Dalmia, Jure Leskovec, Cyril Zakka, Eduardo Pontes Reis, and Pranav Rajpurkar. Med-Flamingo: a multimodal medical few-shot learner. In *Proceedings of the 3rd Machine Learning for Health Symposium*, volume 225 of *Proceedings of Machine Learning Research*, pp. 353–367. PMLR, 2023. URL https://proceedings.mlr.press/v225/moor23a.html.

Jayneel Parekh, Pegah Khayatan, Mustafa Shukor, Arnaud Dapogny, Alasdair Newson, and Matthieu Cord. Learning to steer: Input-dependent steering for multimodal LLMs. In *Advances in Neural Information Processing Systems*, volume 38, pp. 159799–159834, 2025. doi: 10.52202/085713-5341.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.06295.

Weixuan Wang, Jingyuan Yang, and Wei Peng. Semantics-adaptive activation intervention for LLMs via dynamic steering vectors. In *International Conference on Learning Representations*, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c4d26a95fd83f8e590f81c54ae670b5d-Abstract-Conference.html.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency, 2025b. URL https://arxiv.org/abs/2508.18265.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. ReFT: Representation finetuning for language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 63908–63962, 2024. doi: 10.52202/079017-2041.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. AxBench: Steering LLMs? Even simple baselines outperform sparse autoencoders. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 67035–67080. PMLR, 2025. URL https://proceedings.mlr.press/v267/wu25a.html.

Wannan Yang, Xinchi Qiu, Lei Yu, Yuchen Zhang, Aobo Yang, Narine Kokhlikyan, Nicola Cancedda, and Diego Garcia-Olano. Hallucination reduction with CASAL: Contrastive activation steering for amortized learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/11fa9df8d9c7260b04bbab1619d0a869-Abstract-Conference.html.

A IMPLEMENTATION DETAILS

Table 6: PCS-AC configurations. Layer indices are zero based. Text strength is development-selected; the primary VLM comparison uses full strength.

Setting	Qwen2.5-VL-7B	InternVL3.5-8B	LLaMA-3.1-8B
Hidden dimension / layer	3584 / 14	4096 / 18	4096 / 2
Subspace dimension / rank	64 / 8	64 / 8	64 / 8
Radius fraction ρ / strength s	.1 / 1	.1 / 1	.1 / .75
Refinement mode	full controller	full controller	output head
Gradient-updated parameters	29,248	33,344	576
Fixed basis scalar entries	229,376	262,144	262,144
Frozen input-matrix entries	0	0	32,768
Source / development examples	3841 / 647	3841 / 647	512 / 600
Native-correct source $\lvert\mathcal{K}_0\rvert$	2417	1713	483
Margin scale γ (rounded)	1.608326	2.097795	2.878519
Epochs / optimizer updates	2 / 962	2 / 962	8 / 512
Keep λ / ridge η	1 / .01	1 / .01	1 / .01
Learning rate / batch size	.0005 / 8	.0005 / 8	.002 / 8
Seeds / evaluated checkpoint	17, 29 / final	17, 29 / final	17, 29 / final

Optimization and intervention. We use AdamW with constant learning rate, zero weight decay, betas $(.9, .999)$, epsilon 10^{-8}, and gradient norm clipping at one. The ridge penalty in Eq. 5 applies only to compilation. Updates average per-example gradients over the actual batch size, including the final incomplete VLM batch; each epoch uses a seed-dependent source permutation. The frozen backbone stays in evaluation mode. Model and controller computations use FP32; spectral and regression solves use FP64 and export FP32 assets. The intervention replaces the last prompt-token state at the specified decoder-block output. Refinement differentiates through the clipped action and frozen downstream layers.

Model-specific fitting and selection. Each backbone fits its own Q, normalization statistics, compiled factors, margin scale, and native-correct population using source examples only. J-PCA was selected against an alternative action-prediction subspace using source out-of-fold evaluation and C before fitting the main controllers. The three folds group dependencies and stratify by environment and label; the final basis and controller use the full source set. Appendix J specifies text hyperparameter and strength selection.

Preservation normalization. The implemented per-example objective is

$$\ell_i = \mathrm{softplus}(-y_i m_i^\theta) + \lambda \frac{N}{|\mathcal{K}_0|} \mathbf{1}\{i \in \mathcal{K}_0\} \frac{[y_i(m_i^0 - m_i^\theta)]_+}{|m_i^0| + \gamma}. \tag{10}$$

Its source average equals Eq. 8. The fixed set $\mathcal{K}_0$ uses the native decision rule, including a positive prediction at zero margin. The Keep hinge uses the zero ReLU subgradient. NoKeep removes only this penalty, retaining the prediction loss and action cap.

Stored assets and inference cost. Deployment stores Q, W_{in}, W_{out}, b, μ, and σ: $dk + r(d + k) + k + d + 1$ floating-point entries, including frozen assets under output-head refinement. Matrix products cost $O(dr + rk + dk)$ per example within one backbone forward pass. Offline construction computes native margin gradients, and refinement backpropagates through the frozen downstream layers.

B WHY THE LOCAL TEACHER HAS THIS FORM

For a proposed coefficient vector c, the first-order signed margin change is $y_i g_i^\top c$. Cauchy–Schwarz bounds it by $\|g_i\|_2 \|c\|_2$, with equality in the direction $y_i g_i$. Because $Q^\top Q = I$, the hidden-state

norm $\|Qc\|_2$ equals $\|c\|_2$. For $g_i \neq 0$, a reachable gap Δ_i therefore yields

$$\arg\min_c \left\{ \tfrac{1}{2} \|c\|_2^2 : y_i g_i^\top c \geq \Delta_i,\ \|c\|_2 \leq R_i \right\} = y_i \frac{\Delta_i}{\|g_i\|_2^2} g_i. \tag{11}$$

When $\Delta_i > R_i \|g_i\|_2$, the constraints are infeasible. Maximizing the attainable signed gain over the ball instead gives $c = y_i R_i g_i / \|g_i\|_2$. These two cases produce Eq. 4. A zero gap requires no action, while a zero response admits no first-order gain. The implementation treats response norms at most 10^{-12} as zero. This optimum applies to the first-order approximation.

C Normalization and the reduced-rank compiler

The controller uses samplewise RMS normalization followed by source centering and one global scalar scale:

$$u(h) = \frac{h}{\sqrt{\max(d^{-1}\|h\|_2^2, 10^{-12})}}, \quad \mu = \frac{1}{N}\sum_i u(h_i), \quad \sigma = \sqrt{\frac{1}{Nd}\sum_i \|u(h_i) - \mu\|_2^2}. \tag{12}$$

We require $\sigma > 10^{-8}$ and a nonempty native-correct set. The movement radius uses the original h before normalization.

To solve Eq. 5, stack $x(h_i)^\top$ and $(c_i^*)^\top$ into $X \in \mathbb{R}^{N\times d}$ and $C \in \mathbb{R}^{N\times k}$, and center their columns to obtain $\widetilde{X}, \widetilde{C}$. Define

$$A = \frac{\widetilde{X}^\top \widetilde{X}}{N} + \eta I_d, \qquad T = \frac{\widetilde{X}^\top \widetilde{C}}{N}, \qquad F = A^{-1}T. \tag{13}$$

Let V_r contain the leading r eigenvectors of $T^\top A^{-1} T$. The solution and bias are

$$W_0 = (F V_r V_r^\top)^\top, \qquad b_0 = \bar{c} - W_0 \bar{x}. \tag{14}$$

Indeed, eliminating the unpenalized intercept and writing $B = W^\top$ turns the objective into

$$\frac{\|\widetilde{C}\|_F^2}{N} - \|A^{-1/2}T\|_F^2 + \|A^{1/2}B - A^{-1/2}T\|_F^2. \tag{15}$$

Since A is positive definite, the rank constraint is preserved by multiplication by $A^{1/2}$. The best rank-r approximation keeps the leading right singular subspace of $A^{-1/2}T$, giving Eq. 14. We compute F with a Cholesky solve. For a compact SVD $W_0 = U\Sigma V^\top$, initialize $W_{\text{out}} = U\Sigma^{1/2}$ and $W_{\text{in}} = \Sigma^{1/2}V^\top$. This solves the unclipped regression objective exactly; the nonlinear refinement follows afterward.

Initialization ablations. The VLM zero-output control retains Q and source statistics, initializes W_{in} with a seeded Gaussian matrix scaled by $1/\sqrt{d}$, and zeros W_{out} and b. The text control instead retains the frozen *compiled* W_{in} and zeros only the output map and bias. Alternative-basis comparisons regenerate teacher targets and the regression solution for each basis.

D Evaluation cohorts and selection details

Table 7 summarizes the experimental cohorts. RadImageNet-VQA supplies binary abnormality questions, VQA-RAD tests transfer to new images and question formulations, and our synthetic text collection tests assertion decisions without images. Counts refer to the subsets used here.

Visual data. RadImageNet-VQA (Butsanets et al., 2026) supplies the labels and images for our source, development, and evaluation cohorts, all drawn from publisher training shards. VQA-RAD contributes the 251 binary questions in its original test split. The new-environment protocol was fixed before its first evaluation; subsequent inspection makes this analysis retrospective.

Source, development, and evaluation cohorts share no sample IDs, known dependency groups, source-image hashes, or decoded-pixel hashes. Text splits similarly share no IDs, known groups, or text–entity hashes. These comparisons cover known dependencies and exact duplicates; group counts do not establish patient independence.

Table 7: Datasets and cohort roles used in this study. Counts are QA or assertion examples and known dependency groups, not patients.

Dataset / task	Experimental role	Examples	Groups	Split / provenance
RadImageNet-VQA Image + text: abnormality	Source fitting	3,841	3,841	Publisher train; seven source environments.
	Development (C)	647	643	Project split; source environments.
	Same-domain evaluation	2,100	2,100	Project holdout from publisher train.
	New-environment evaluation	1,841	1,809	Project holdout; CT lung and MRI hip.
	Target support (all blocks)	192	192	Three blocks per environment; 32 representatives each.
	Common target query	1,647	1,617	New environments, excluding all support groups.
VQA-RAD Image + text: radiology QA	Cross-dataset evaluation	251	135/133	Original binary test; grouping versions noted below.
	Adaptation support (all blocks)	64	64	Original train; two blocks, 32 labels per deployment.
	Adaptation development	131	37	Original train; shared residual-cap selection.
Project assertions Text only: Present/Absent	Source fitting	512	512	Controlled synthetic data; balanced source.
	Development	600	600	Separate synthetic development split.
	Retrospective evaluation	3,360	3,360	Four observed subsets, 840 assertions each.

RadImageNet-VQA (Butsanets et al., 2026) and VQA-RAD (Lau et al., 2018) supply visual QA; we constructed the synthetic text data. The 1,841-QA new-environment cohort is divided into 192 support QA, 2 same-group exclusions, and 1,647 query QA. Each deployment uses 8, 16, or 32 labels per environment from one block. VQA support/development exclude dependencies on all official test questions. The stricter grouping used for adaptation yields 133 test groups with the same test membership (Appendix N). All evaluations are retrospective.

Text data. We use a controlled Present/Absent collection constructed for our earlier experiments. Each example pairs a statement with a target entity and a binary label. We sampled 512 source examples, 256 per class, from a 1,586-example pool using seed 17. The development set contains 600 examples; evaluation combines subsets V2, V3, V5, and V6, each with 840 examples. Both the development set and each evaluation subset are label-balanced. The texts and labels are unchanged. The original synthetic generator has not been fully reconstructed.

Target-support protocol. For each of CT lung and MRI hip, we reserve three disjoint blocks of 32 representative support groups; 8- and 16-label budgets are nested prefixes. Excluding all 192 representatives and two additional QA in their groups leaves a common query of 1,647 QA: 723 lung and 924 hip. Each deployment uses one block per environment, at most 64 labels in total. Strength is selected independently for each seed and environment using only that block. Results average two seeds and three blocks on this shared query.

E Prompts, candidate scoring and inference configuration

Visual interface. Both VLMs receive one user message with the image followed by this text:

```
{question}
Answer with exactly one of:  yes, no.
```

We apply the processor's chat template with `add_generation_prompt=True`. At the final prompt position, the margin is the next-token log-probability of `yes` minus that of `no`; the token IDs are 9693 and 2152 for both VLM tokenizers. Zero margin predicts `yes`. This is restricted candidate scoring without autoregressive answer generation.

Qwen uses processor-defined dynamic resizing with pixel bounds of 3,136 and 200,704 (`min_pixels` and `max_pixels`). InternVL uses one 448×448 image tile, with `crop_to_patches=False` on VQA-RAD. Native and intervened models use the same preprocessing.

Text interface. The text experiment uses a plain tokenized prompt, without a chat template or truncation:

```
Statement: {text}
Entity: {target_entity}
Is the entity Present or Absent?
Answer:
```

The next-token margin is the logit of `Present` (token 21886) minus that of `Absent` (token 81351), with ties predicting `Present`. All interventions act on the final prompt-token hidden state at the layers in Section 5.

Versions and environment. The implementation specifies model revisions and the six training configurations. We use Python 3.12.3, PyTorch 2.7.1+cu126, Transformers 4.57.1, Tokenizers 0.22.1, and NumPy 1.26.4. Main experiments use FP32 with TF32 disabled.

F Shared-direction and source-calibration controls

Action and matched refinement. The shared-direction control replaces conditional coordinates with a learned vector $b \in \mathbb{R}^{64}$:

$$\delta_{\text{shared}}(h; s) = s \, \text{clip}_{0.1\|h\|_2}(Qb). \tag{16}$$

It retains the source-fitted J-PCA64 basis and initializes b to the mean of the same teacher coordinates c_i^*. The direction is shared; the clipping radius and downstream response still depend on the input. Only the 64 entries of b are trained, versus 29,248 PCS-AC parameters on Qwen and 33,344 on InternVL.

Data, seeds, preprocessing, intervention location, and source-example order match the main VLM configuration: 3,841 source QA, C647, a 2,100-QA holdout, two epochs, and 962 updates. We use the optimizer, batch size, BCE-plus-Keep objective, and preservation statistics in Appendix A. Parameterization and initialization differ as specified above.

Auxiliary source-C strength selection. For each backbone and method, we choose one strength from $\{0, .125, .25, .5, .75, 1\}$ by summed C correctness across both seeds, breaking ties toward smaller strength. This auxiliary QA-weighted selection uses no evaluation labels. Table 8 reports both fixed and selected strengths. At $s = 1$, balanced accuracy is 55.098% versus 74.039% for shared direction versus PCS-AC on Qwen, and 63.734% versus 83.695% on InternVL.

Native threshold calibration. The Native threshold in Table 3(b) is selected only on C by the group-weighted objective, using damage and threshold to break ties, without changing hidden states. The holdout has 1,453 positive and 647 negative labels. Native balanced accuracy is 53.591% on Qwen and 55.241% on InternVL, versus 74.039% and 83.695% for PCS-AC.

G Action-space construction across backbones

Construction and refinement. Table 4(a) compares a random orthogonal basis, centered PCA of raw hidden states (H-PCA), and budget-weighted, uncentered margin-Jacobian PCA (J-PCA). Each 64-dimensional basis refits its teacher targets and rank-eight compiler with ridge coefficient .01. Refinement updates all factors: 29,248 parameters on Qwen and 33,344 on InternVL. Capacity

Table 8: Source-development strength check for the shared-direction control. Accuracy (%) is measured on the 2,100-QA same-domain holdout. C-selected rows use the auxiliary selection protocol in Appendix F.

Configuration	Qwen	InternVL
Native	68.762	38.952
LoReFT, fixed configuration	69.762	73.881
Shared direction, $s = 1$	67.643	53.167
Shared direction, C-selected s	68.024	53.167
PCS-AC (Ours), $s = 1$	**80.952**	**81.619**
PCS-AC (Ours), C-selected s	**80.952**	**81.619**

C selects $s = .5$ for Qwen's shared direction and $s = 1$ otherwise. Learned methods report two-seed means; bold marks the highest listed accuracy per model.

matches within each backbone, while basis, teacher targets, and initialization change together. One random-basis draw per backbone is shared across its two training seeds.

All pipelines use the main VLM settings in Appendix A, paired training-example order, the same 2,100-QA holdout, and $s = 1$. An auxiliary InternVL check selects strength on C647 over $\{0, .125, .25, .5, .75, 1\}$ by summed correctness across both seeds, with ties favoring smaller strength. All three spaces select $s = 1$.

Repairs, damage, and class balance. Table 9 reports InternVL repairs, damage, and balanced accuracy. Relative to H-PCA, J-PCA adds 812.5 repairs and 57 damaged decisions, yielding 755.5 additional net corrections. Relative to fixed LoReFT, it adds 162 repairs and reduces damage by 0.5. These counts explain the accuracy gains in Section 5.5.

Table 9: InternVL action-space comparisons on the 2,100-QA same-domain holdout. Accuracy and balanced accuracy (BA) are percentages; repair, damage, and net correction are mean example counts relative to Native. All interventions use $s = 1$.

Configuration	Accuracy	BA	Repair	Damage	Net
Native	38.95	55.24	0	0	0
LoReFT (fixed)	73.88	77.89	798.0	64.5	733.5
Random orthogonal	41.76	57.19	65.5	6.5	59.0
Activation PCA	45.64	59.86	147.5	7.0	140.5
J-PCA (Ours)	**81.62**	**83.69**	960.0	64.0	**896.0**

Net correction equals repair minus damage. All learned-method entries average two training runs; Native is deterministic. Capacity matches across the three action-space configurations within InternVL (33,344 parameters each). The fixed LoReFT reference uses its separately specified architecture and learning rate. Bold marks the highest accuracy, BA, and net correction.

Environment coverage. Table 10 partitions the same holdout into seven environments of 300 QA each. J-PCA exceeds both alternative spaces throughout and fixed LoReFT in six environments; MRI shoulder favors LoReFT. Learned-method entries average two training runs.

Table 10: InternVL action-space accuracy (%) across all seven same-domain environments. Each environment contains 300 QA. J-PCA exceeds both alternative action-space constructions throughout; fixed LoReFT remains higher on MRI shoulder.

Environment	Native	LoReFT	Random	H-PCA	J-PCA (Ours)
CT abdomen	52.00	60.00	53.17	53.67	**63.83**
MRI abdomen	80.33	80.83	80.67	80.83	**82.17**
MRI ankle/foot	20.33	70.33	24.33	28.67	**81.50**
MRI brain	65.67	78.17	66.50	67.33	**86.33**
MRI knee	17.00	84.00	21.67	28.50	**87.17**
MRI shoulder	22.33	**92.33**	31.00	45.50	89.50
MRI spine	15.00	51.50	15.00	15.00	**80.83**
Overall	38.95	73.88	41.76	45.64	**81.62**

All interventions use $s = 1$, the common candidate-scoring interface, and zero decision threshold. H-PCA denotes activation PCA. Overall pools predictions across the seven environments. Bold marks the highest listed accuracy in each row.

H Full frozen-transfer results

Table 11: Frozen transfer: Native versus PCS-AC at $s = 1$, with no target labels used for strength selection. Δ is the change from Native in percentage points.

Model	Target cohort	Native	PCS-AC (Ours)	Δ
Qwen	CT lung (821)	64.56	**68.21**	+3.65
Qwen	MRI hip (1,020)	**80.00**	63.04	-16.96
Qwen	Pooled new env. (1,841)	**73.11**	65.34	-7.77
Qwen	VQA-RAD (251)	**78.88**	74.70	-4.18
InternVL	CT lung (821)	64.92	**75.15**	+10.23
InternVL	MRI hip (1,020)	21.76	**47.65**	+25.88
InternVL	Pooled new env. (1,841)	41.01	**59.91**	+18.90
InternVL	VQA-RAD (251)	**78.49**	71.51	-6.97

The pooled new-environment row combines lung and hip. VQA-RAD uses its original binary test subset.

Table 11 reports source-frozen transfer at $s = 1$, without target labels. Qwen improves on CT lung but declines on MRI hip; InternVL improves on both RI environments. Both models decline on VQA-RAD. Source-C selection also chooses $s = 1$, reproducing these results.

The pooled RI row combines lung and hip. Support adaptation in Table 12 uses a smaller 1,647-QA query after excluding all support groups; its Native score therefore differs from the full 1,841-QA cohort.

H.1 Strength selection with target support

Controllers remain frozen while 8, 16, or 32 support labels per environment select strength from $\{0, .125, .25, .5, .75, 1\}$. Selection maximizes support correctness, then minimizes damage, then prefers smaller strength. Three disjoint support blocks define nested budgets, evaluated on the same 1,647-QA query excluding all reserved support groups (Appendix D). At 32 labels per environment, Qwen reaches 74.44%, up from 65.42% at fixed strength, while damage falls from 304.5 to 37.5. InternVL reaches 59.22%, above Native's 40.74% but below fixed-strength PCS-AC's 60.02%. Appendix N evaluates residual fitting with additional target supervision.

I Compilation and initialization

Table 4(b) separates compilation, refinement, and output initialization. Compiled initialization improves refined VLM accuracy by 1.79 and 1.07 points over zero-output controls. Text favors zero output initialization (95.86% versus 95.03%); that control retains the frozen compiled input map,

Table 12: Selecting deployment strength with target support. Accuracy and change from Native (Δ, percentage points) on the common 1,647-QA query. The label budget is shown for each setting; controller parameters remain fixed.

		Qwen		InternVL	
Configuration	Labels/env.	Acc.	Δ	Acc.	Δ
Native	0	73.28	+0.00	40.74	+0.00
PCS-AC (Ours), fixed $s = 1$	0	65.42	-7.86	**60.02**	+19.28
PCS-AC (Ours), support-selected	8	71.95	-1.34	48.14	+7.40
PCS-AC (Ours), support-selected	16	73.26	-0.02	55.11	+14.37
PCS-AC (Ours), support-selected	32	**74.44**	+1.15	59.22	+18.48

Support-selected entries average three support blocks and two seeds, using 8/16/32 labels per environment per deployment. Source-C selection also chooses $s = 1$. Bold marks the highest accuracy across the listed budgets. The common query is the 1,841-QA cohort from Table 11 after excluding all support groups.

whereas VLM controls use random trainable input maps. The text result therefore concerns output initialization, not removal of compilation. Appendix C gives the exact constructions.

J TEXT REFINEMENT AND DEVELOPMENT SELECTION

The text controller independently fits a LLaMA response basis and compiler on the source set in Appendix D, then refines only the output map and bias. The mean-difference reference uses the source Absent-minus-Present hidden-state direction, the same relative movement budget, and development-selected strength. Table 13 reports component accuracies and paired corrections.

Table 13: Text accuracy and paired correction counts for the selected configuration and component ablations.

Method	Accuracy	Repair	Damage	Net
Native	89.05	0	0	0
Mean-difference steering	89.02	0	1	-1
Compiled only	93.12	174	37	137
LoReFT reference (2 epochs)	93.94	213.5	49	164.5
PCS-AC (Ours, 8 epochs)	95.03	250.5	49.5	201
PCS-AC without Keep	95.06	250.5	48.5	202
PCS-AC with zero output init.	**95.86**	272	43	229

Refined PCS variants use $s = .75$; the reference, compiled-only and mean-difference interventions use development-selected $s = 1$. The zero-output variant retains the compiled input map; it is evaluated as a separate ablation.

Table 14: PCS-AC text development search and the fixed LoReFT reference. For each PCS-AC configuration, strength maximizes summed development correctness across both seeds, with ties favoring the smaller strength.

Family	Learning rate	Epochs	s	Dev (600)	Evaluation (3,360)
PCS	0.0005	2	1	96.25	–
PCS	0.002	2	0.75	96.33	–
PCS	0.005	2	0.75	96.75	–
PCS	0.002	4	0.75	96.83	–
PCS	0.002	8	0.75	96.92	95.03
LoReFT (fixed)	0.002	2	1	97.75	93.94

Across PCS-AC configurations, ties prefer fewer epochs, smaller strength, lower learning rate and configuration identifier. Table 2 reports evaluation accuracy for the selected PCS-AC configuration and fixed LoReFT; dashes indicate unreported evaluation scores. LoReFT uses two epochs and $s = 1$.

The PCS-AC search evaluates five learning-rate/epoch configurations at strengths $\{0, .125, .25, .5, .75, 1\}$ using both seeds. Selection maximizes summed development correct-

ness, with ties preferring fewer epochs, smaller strength, lower learning rate, and configuration identifier. Table 14 summarizes development scores. The selected PCS-AC configuration uses eight epochs; fixed LoReFT uses two, so optimization budgets differ.

NoKeep and zero-output ablations match the selected configuration's epochs, learning rate, architecture, and strength. Both are reported as ablations; evaluation scores were not used to choose the main configuration. Table 15 gives their results on all four subsets.

Table 15: Text component ablations across four subsets of 840 examples. All PCS variants use $s = .75$. These subsets were examined during earlier development.

Subset	Native	PCS-AC (Ours)	No Keep	Zero output init.
V2	93.93	99.05	98.93	99.40
V3	88.21	90.48	90.54	92.38
V5	88.10	95.77	95.95	96.07
V6	85.95	94.82	94.82	95.60

K Paired correction and preservation

At matched architecture, initialization, budget, and strength, Keep reduces same-domain damage from 137.5 to 95.5 on Qwen and from 100 to 64 on InternVL. Fewer repairs produce accuracy costs of 0.19 and 2.76 points. Text accuracy is nearly unchanged (95.03% with Keep; 95.06% without). Table 16 reports all main and transfer cohorts as two-seed means. For n examples, let R count Native errors repaired by the intervention and D count Native correct decisions reversed. The identity

$$\mathrm{Acc}_{\mathrm{intervention}} - \mathrm{Acc}_{\mathrm{Native}} = \frac{R - D}{n} \tag{17}$$

holds per seed and for their mean. Keep controls the repair–damage trade-off during source fitting; its effects vary by evaluation cohort, as the full counts show.

Table 16: Keep ablation: accuracy and paired repair/damage counts. Each row holds architecture, refinement budget and strength fixed.

		Without Keep		With Keep	
Model	Evaluation	Acc.	Repair/damage	Acc.	Repair/damage
Qwen	Same domain	81.14	397.5/137.5	80.95	351.5/95.5
Qwen	New environment	62.74	216.5/407.5	65.34	194.5/337.5
Qwen	VQA-RAD	73.11	9/23.5	74.70	13/23.5
InternVL	Same domain	84.38	1054/100	81.62	960/64
InternVL	New environment	60.32	484.5/129	59.91	452/104
InternVL	VQA-RAD	69.72	13/35	71.51	13/30.5
Llama	Text assertions	95.06	250.5/48.5	95.03	250.5/49.5

L Preservation and deployment dose

We compare Keep with reducing deployment strength using the original Keep and NoKeep checkpoints at post-clipping multipliers $s \in \{0, .25, .5, .75, 1\}$. Architecture, initialization, source data, example order, and refinement budget match within each VLM. Table 17 reports every point on the 2,100-QA holdout as a two-seed mean. C647 selects strength by summed correctness across seeds, with ties preferring smaller values; all four model/family pairs select $s = 1$.

Repair–damage trade-off. Keep usually reduces both repairs and damage, with lower accuracy at every nonzero tested strength. InternVL at $s = .25$ has equal damage. Define dominance as at least as many repairs and no more damage, with one strict inequality. No measured Keep point dominates a NoKeep point; NoKeep dominates Keep at InternVL $s = .25$. The curves therefore show a conservative correction trade-off, without an advantage over strength reduction.

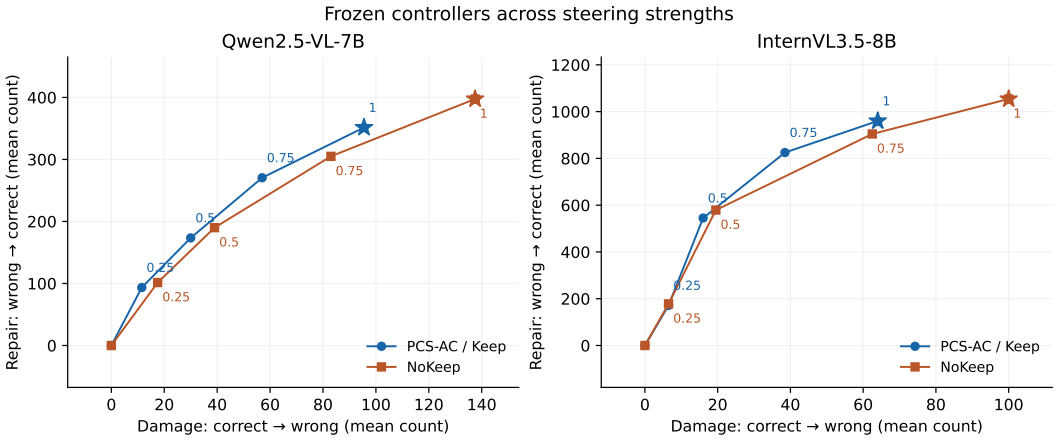


Figure 2: Repair–damage trade-offs for Keep and NoKeep. Each point averages two training seeds on the same 2,100-question project holdout. Labels give the post-clipping strength; stars mark $s = 1$, which source calibration also selects for every family. Lines connect the measured points for readability.

Table 17: Accuracy and paired corrections across strengths on the 2,100-question RadImageNet-VQA holdout. Entries average two training seeds. BA is balanced accuracy; repair and damage count changes relative to Native. Both families recover Native at $s = 0$. Selection on the separate source calibration set gives $s = 1$ for each model and family.

Backbone	Family	s	Acc. (%)	BA (%)	Repair	Damage	Net
Qwen2.5-VL-7B	Keep	0	68.76	53.59	0.0	0.0	0.0
Qwen2.5-VL-7B	Keep	0.25	72.67	58.41	93.5	11.5	82.0
Qwen2.5-VL-7B	Keep	0.5	75.60	63.31	173.5	30.0	143.5
Qwen2.5-VL-7B	Keep	0.75	78.93	69.49	270.5	57.0	213.5
Qwen2.5-VL-7B	Keep	1	80.95	74.04	351.5	95.5	256.0
Qwen2.5-VL-7B	NoKeep	0	68.76	53.59	0.0	0.0	0.0
Qwen2.5-VL-7B	NoKeep	0.25	72.76	58.58	101.5	17.5	84.0
Qwen2.5-VL-7B	NoKeep	0.5	75.95	63.72	190.0	39.0	151.0
Qwen2.5-VL-7B	NoKeep	0.75	79.33	70.55	305.0	83.0	222.0
Qwen2.5-VL-7B	NoKeep	1	81.14	75.55	397.5	137.5	260.0
InternVL3.5-8B	Keep	0	38.95	55.24	0.0	0.0	0.0
InternVL3.5-8B	Keep	0.25	46.81	60.83	171.5	6.5	165.0
InternVL3.5-8B	Keep	0.5	64.14	73.08	545.0	16.0	529.0
InternVL3.5-8B	Keep	0.75	76.40	81.02	825.0	38.5	786.5
InternVL3.5-8B	Keep	1	81.62	83.69	960.0	64.0	896.0
InternVL3.5-8B	NoKeep	0	38.95	55.24	0.0	0.0	0.0
InternVL3.5-8B	NoKeep	0.25	47.12	61.01	178.0	6.5	171.5
InternVL3.5-8B	NoKeep	0.5	65.60	73.89	579.0	19.5	559.5
InternVL3.5-8B	NoKeep	0.75	79.05	81.90	904.5	62.5	842.0
InternVL3.5-8B	NoKeep	1	84.38	84.21	1054.0	100.0	954.0

M DEPLOYMENT TIMING PROTOCOL

We compare the fitted PCS-AC controller and fixed LoReFT with Native on all three backbones. We use one checkpoint per trained intervention and 96 source-development inputs per backbone, selected by a label-blind hash. We run all three methods on every input for three repetitions, balancing the six possible method orders. The protocol uses 2,592 timed forwards and 78 warm-up and validation forwards.

Timing setup. Inference uses an exclusive NVIDIA A100-SXM4-40GB GPU, batch size one, FP32, TF32 disabled, and eager execution. CPU wall-clock timing synchronizes CUDA at both

boundaries. Model timing starts with device-resident inputs. End-to-end timing starts from decoded in-memory inputs and includes preprocessing, device transfer, and forward computation; it excludes file I/O, decoding, and model/controller loading. Native has no intervention hook. The timed interventions include their numerical checks; disk writes occur outside the timed interval.

Paired overhead. For input i and repetition r, let t_{ir}^{M} be forward time under method M. Table 18 reports $\mathrm{median}_{i,r}(t_{ir}^{M} - t_{ir}^{\mathrm{Native}})$ and $\mathrm{median}_{i,r}[100(t_{ir}^{M}/t_{ir}^{\mathrm{Native}} - 1)]$, computing differences and ratios before aggregation. Table 19 gives absolute model and end-to-end times.

Table 18: Measured deployment overhead for binary-decision prefill, batch size one, FP32. Each backbone uses 96 development inputs and three repetitions per method. For extra latency and relative overhead, we compute differences and ratios against Native for each input and repetition, then take the median. Fixed LoReFT uses the original comparison configuration.

Backbone	Native (ms)	PCS Δ (ms)	PCS (%)	LoReFT Δ (ms)	LoReFT (%)
Qwen2.5-VL-7B	224.16	+0.78	0.36	+1.65	0.71
InternVL3.5-8B	698.18	+0.50	0.07	+0.50	0.07
LLaMA-3.1-8B	151.48	+5.18	4.04	+7.29	5.76

Table 19: Absolute timing statistics in milliseconds. Model timing starts with inputs on the GPU. End-to-end timing starts from decoded in-memory inputs and includes preprocessing, transfer, and the model forward pass; it excludes file I/O, image decoding, and model loading. Table 18 reports paired overhead computed before aggregation.

Backbone	Method	Model median	Model p90	End-to-end median
Qwen2.5-VL-7B	Native	224.16	261.55	228.57
Qwen2.5-VL-7B	PCS	220.15	260.82	225.85
Qwen2.5-VL-7B	LoReFT	224.92	265.25	229.16
InternVL3.5-8B	Native	698.18	732.33	715.88
InternVL3.5-8B	PCS	698.94	732.55	715.98
InternVL3.5-8B	LoReFT	696.86	733.57	713.20
LLaMA-3.1-8B	Native	151.48	203.06	154.55
LLaMA-3.1-8B	PCS	157.67	208.04	161.76
LLaMA-3.1-8B	LoReFT	165.10	213.56	168.46

These measurements cover binary-decision prefill. All controllers reside in the same backbone process, so peak memory does not isolate controller-loading cost. Appendix A accounts for stored assets and offline fitting costs.

N Exploratory target adaptation beyond scalar strength

A bounded residual head. This exploratory extension adds a zero-initialized residual output map and bias to the support-strength controller. The backbone, 64-dimensional basis, normalization, and compiled input map remain frozen. Its 576 parameters fit 32 support labels for eight epochs, with batch size eight, learning rate .002, prediction-plus-Keep loss of weight one, and final-checkpoint evaluation.

Let $z(h)$ denote the existing rank-eight input features and $\delta_0(h)$ the source controller's already clipped action. The adapted action is

$$\delta_{\mathrm{adapt}}(h) = \mathrm{clip}_{.1\|h\|_2}\left[s_{\mathrm{support}}\delta_0(h) + \mathrm{clip}_{\rho_r\|h\|_2}(Q[W_r z(h) + b_r])\right]. \tag{18}$$

Residual budgets are $\rho_r = .02$ (R02) and .05 (R05), with total cap $.1\,\|h\|_2$. Zero residual recovers the support-strength controller. Unlike scalar adaptation, R02/R05 train target-specific parameters.

New-environment query and simple controls. RI uses the common 1,647-QA query and three support blocks from Appendix D, fitting a separate head with 32 labels for each of lung and hip.

Table 20: Exploratory target adaptation: accuracy (%) and target-label budgets. All listed methods within each panel use the same evaluation examples. Bold marks the highest accuracy in each backbone column, including ties. Target-label budgets are shown separately.

	Target labels / deployment			
Method	Support	Shared dev	Qwen	InternVL
RadImageNet-VQA: common new-environment query (n = 1,647)				
Native	0	0	73.28	40.74
LoReFT, fixed	0	0	74.50	61.29
PCS-AC (Ours), fixed	0	0	65.42	60.02
Support majority	64	0	**76.93**	**76.93**
Native, support threshold	64	0	75.21	76.62
LoReFT, support strength	64	0	73.74	61.75
PCS-AC (Ours), support strength	64	0	74.44	59.22
PCS-AC (Ours) + R02	64	0	75.38	61.27
PCS-AC (Ours) + R05	64	0	75.90	62.55
VQA-RAD: official binary test subset (n = 251)				
Native	0	0	**78.88**	**78.49**
LoReFT, fixed	0	0	77.49	68.13
PCS-AC (Ours), fixed	0	0	74.70	71.51
Support majority	32	0	52.99	52.99
Native, support threshold	32	0	77.89	76.49
LoReFT, support strength	32	0	78.69	**78.49**
PCS-AC (Ours), support strength	32	0	77.49	**78.49**
PCS-AC (Ours) + R02	32	131	77.39	76.10
PCS-AC (Ours) + R05	32	131	77.89	75.90

RI uses 32 support labels per environment (64 total), three support blocks, and two seeds. Both residual caps were specified before evaluation and are reported without query-based selection. VQA uses 32 support labels, two blocks, and two seeds; 131 shared development labels select R05. Fixed controllers have the same predictions across support blocks. These retrospective comparisons share their evaluation examples.

Both budgets are evaluated across all blocks and seeds, without selection on query labels. Table 20 includes support-majority, Native-threshold, and intervention controls.

R05 reaches 75.90% on Qwen and 62.55% on InternVL, improving scalar adaptation by 1.46 and 3.33 points. Repairs/damage change from 56.5/37.5 to 76.17/33.17 on Qwen and from 383.67/79.33 to 430.5/71.33 on InternVL. Support-majority prediction remains higher at 76.93%; the support-selected Native threshold also exceeds R05 on InternVL (76.62%).

Separate development selection on VQA-RAD. VQA-RAD support and development come from its original training split. Excluding dependencies on all 451 official test questions leaves 384 of 942 binary training QA in 101 groups. Grouping uses identical pixels, dHash distance at most two, and known MedPix case URLs. A label-blind hash reserves two disjoint 32-QA support blocks and 131 development QA in 37 groups; 189 additional QA from support groups are unused. The 251 binary test QA form 133 groups under this stricter rule, versus the original 135.

Each deployment uses 32 support labels *plus* 131 shared development labels. Cap selection maximizes development accuracy averaged across models, seeds, and blocks, breaking ties by damage and cap size. Selection favors R05 (81.775% versus 81.679% for R02); both caps are reported on the previously observed test subset.

R05 reaches 77.89% and 75.90% on Qwen and InternVL, below Native's 78.88% and 78.49%. Scalar adaptation matches Native on InternVL; residual fitting loses 2.59 points. The gains observed on RI therefore do not extend to VQA-RAD.

Source-trained gate. A source-C gate takes Native and full-controller margins as inputs. Qwen/InternVL accuracies are 65.42/60.14% on RI (from 65.42/60.02%) and unchanged at 74.70/71.51% on VQA-RAD. This two- or three-forward gate is excluded from main timings.

O QUALITATIVE EXAMPLES OF CORRECTION AND PRESERVATION

We show eight visual and four text examples selected by outcome from the cohorts in Tables 1 and 2. They cover all seven visual environments and four text subsets, using the main configurations. Trained-method predictions agree across seeds 17 and 29. "Correct" means agreement with the dataset label.

O.1 VISUAL CORRECTIONS WITH QWEN2.5-VL-7B

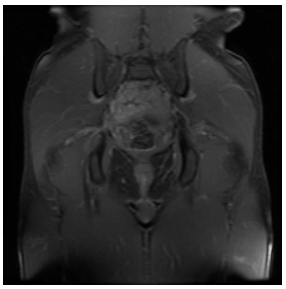
V01: CT abdomen



Dataset label: no

Method	Prediction
Native	yes
LoReFT (fixed)	yes
PCS-AC (Ours)	no
Independent readout	no

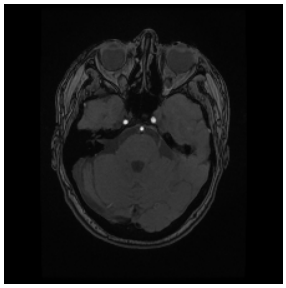
V02: MRI abdomen



Dataset label: no

Method	Prediction
Native	yes
LoReFT (fixed)	yes
PCS-AC (Ours)	no
Independent readout	no

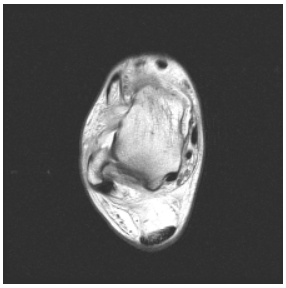
V03: MRI brain



Dataset label: no

Method	Prediction
Native	yes
LoReFT (fixed)	yes
PCS-AC (Ours)	no
Independent readout	no

V04: MRI ankle/foot



Dataset label: yes

Method	Prediction
Native	no
LoReFT (fixed)	no
PCS-AC (Ours)	yes
Independent readout	yes

Figure 3: Selected Qwen corrections (V01–V04). Each evaluated question is "Does this scan reveal any abnormalities?" V01–V03 correct positive predictions to the negative dataset label; V04 corrects a negative prediction to the positive label. Both fixed LoReFT runs retain the Native error in every case. The independent logistic readout also matches every displayed label. Images are reproduced from the evaluated bytes without cropping or added annotations.

O.2 VISUAL CORRECTIONS WITH INTERNVL3.5-8B

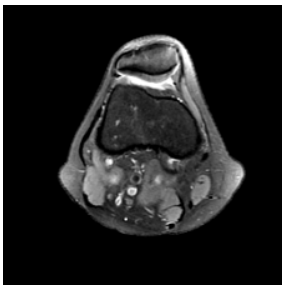
V05: CT abdomen



Dataset label: yes

Method	Prediction
Native	no
LoReFT (fixed)	no
PCS-AC (Ours)	yes
Independent readout	yes

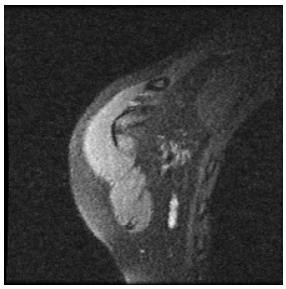
V06: MRI knee



Dataset label: yes

Method	Prediction
Native	no
LoReFT (fixed)	no
PCS-AC (Ours)	yes
Independent readout	yes

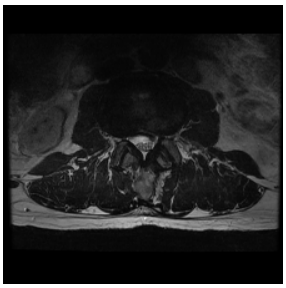
V07: MRI shoulder



Dataset label: no

Method	Prediction
Native	yes
LoReFT (fixed)	yes
PCS-AC (Ours)	no
Independent readout	no

V08: MRI spine



Dataset label: yes

Method	Prediction
Native	no
LoReFT (fixed)	no
PCS-AC (Ours)	yes
Independent readout	yes

Figure 4: Selected InternVL corrections (V05–V08). V05, V06, and V08 use "Does this scan reveal any abnormalities?"; V07 uses "Is this image abnormal?" V05, V06, and V08 correct negative predictions to positive dataset labels; V07 corrects a positive prediction to the negative label. Both fixed LoReFT runs retain the Native error, while the independent readout also matches the label in every case. Images are unchanged from the evaluated source bytes.

O.3 TEXT ASSERTION CORRECTION AND PRESERVATION

T01: V2

Statement: The current description argues away from pleural effusion pneumothorax.

Target: pleural effusion pneumothorax
Dataset label: Absent

Method	Prediction
Native	Absent
Mean-difference	Absent
LoReFT (fixed)	Present
PCS-AC (Ours)	Absent

PCS-AC retains the Native-correct absence judgment; the fixed LoReFT reference reverses it.

T02: V3

Statement: The wording separates absence of granuloma from continued support for rib fracture.

Target: rib fracture
Dataset label: Present

Method	Prediction
Native	Absent
Mean-difference	Absent
LoReFT (fixed)	Absent
PCS-AC (Ours)	Present

The absent distractor is granuloma; the target rib fracture has continued support.

T03: V5

Statement: The evidence can indicate effusion while leaving fracture unsupported.

Target: fracture
Dataset label: Absent

Method	Prediction
Native	Present
Mean-difference	Present
LoReFT (fixed)	Present
PCS-AC (Ours)	Absent

The statement supports effusion while leaving the target fracture unsupported.

T04: V6

Statement: The sentence distinguishes unsupported pleural effusion pneumothorax from supported free air.

Target: free air
Dataset label: Present

Method	Prediction
Native	Absent
Mean-difference	Absent
LoReFT (fixed)	Absent
PCS-AC (Ours)	Present

The unsupported distractor is separate from the supported target free air.

Figure 5: Selected LLaMA-3.1-8B assertion examples (T01–T04), with verbatim statements and target entities. T02–T04 repair errors shared by Native, mean-difference steering, and the fixed LoReFT reference. T01 retains a Native-correct decision that fixed LoReFT reverses. All displayed trained-method labels agree across both seeds. The accompanying explanations describe the synthetic statements; they are written by the authors.

Case selection. Eleven cases illustrate repair; T01 illustrates preservation. We selected repairs where both PCS-AC runs match the dataset label and Native and both fixed LoReFT runs are wrong: 285/2,100 Qwen, 66/2,100 InternVL, and 27/3,360 text examples qualify. Visual selection uses model/environment/label strata and a fixed salted SHA256 ordering. Five earlier CT-negative candidates were excluded from display for field-of-view, environment-match, or readability concerns; all remain in the aggregate evaluation. Other visual strata use their first hash-ranked candidates. Text cases were selected manually after inspecting outcomes and wording.

These examples were chosen to illustrate successful corrections. Aggregate accuracy and Keep effects are measured in the quantitative experiments. Visual labels and environment names follow publisher metadata, without new clinical annotation; Appendix D describes provenance.