UNVEILING ADVERSARIAL PATCHES: PROTOTYPE-GUIDED DETECTION AND LOCALIZATION

Anonymous authors
Paper under double-blind review

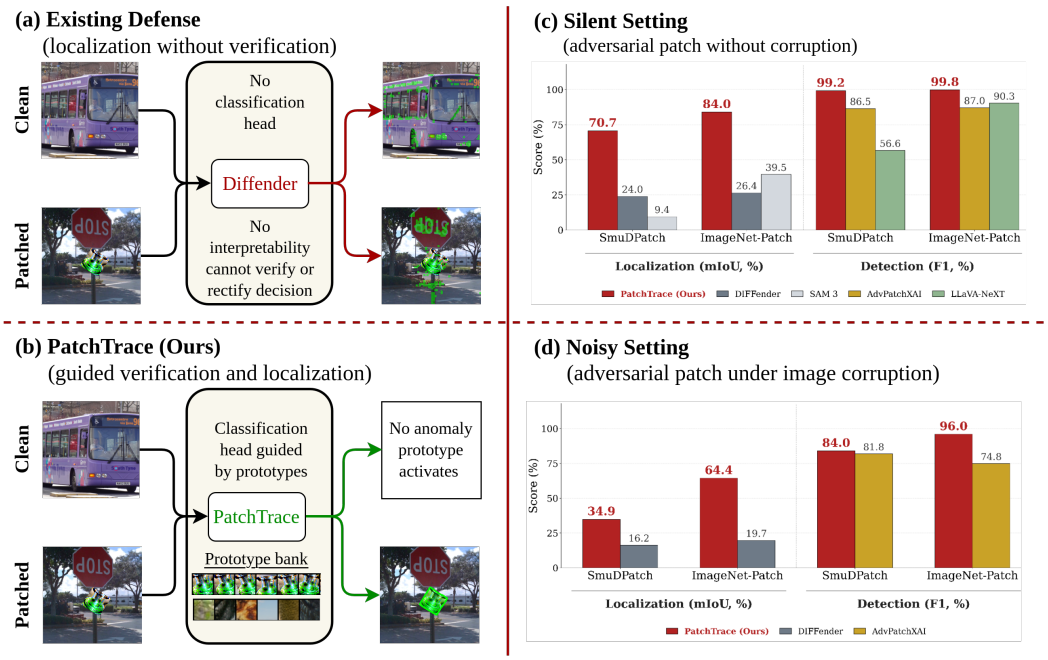


Figure 1: **Why PatchTrace?** *Left:* Existing defenses such as DIFFender (Wei et al., 2025) localize suspicious regions without first verifying whether an attack exists, which can trigger false patch predictions on clean images. PatchTrace instead uses clean and anomaly prototypes to guide classification, suppress anomaly activation for clean inputs, and localize patches only when prototype evidence indicates an attack. *Right:* Under both *silent* (patch-only) and *noisy* (patch + corruption) settings, PatchTrace consistently improves localization and detection over DIFFender (Wei et al., 2025) and AdvPatchXAI (Kumar & Agarwal, 2025). In the silent setting, it also surpasses the foundation model SAM 3 (Carion et al., 2026) for localization and the vision–language model LLaVA-NeXT (Liu et al., 2024) for detection, demonstrating the benefit of prototype-guided, task-specific reasoning for reliable adversarial patch defense.

ABSTRACT

Adversarial patch attacks imperil modern vision systems because they are localized, physically realizable, and highly transferable. Existing defenses typically target either global-level detection (clean vs. patch) or patch localization, but most operate as black boxes and provide limited interpretability. We propose PatchTrace, a prototype-based adversarial patch defense that unifies global attack detection and patch localization within a single interpretable framework. PatchTrace learns to separate clean and anomaly prototype banks, detecting attacks by explicitly competing between normal and adversarial evidence. This dual-bank design enables highly accurate global-level detection of unseen patches and, via a novel Prototype Residual Attention Map (PRAM), it yields faithful, dense localization of adversarial regions. Unlike prior prototype-based defenses that remain restricted to global explanations, PatchTrace extends interpretability to the pixel level, supporting a coherent global-to-local defense pipeline. Extensive experiments demonstrate that PatchTrace achieves near-perfect detection on

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.

unseen patches while delivering stronger, interpretable localization than existing defenses, even in noisy settings. We assert that the proposed PatchTrace is the first prototype-driven, unified end-to-end framework that simultaneously achieves these four properties (detection, localization, interpretability, and robustness), and it can pave the way for a unified development of adversarial patch defense approaches in the future.

1 INTRODUCTION

Adversarial patches threaten safety-critical vision systems in autonomous driving, robotics, and surveillance. These localized perturbations can take physical form and transfer across models and environments (Brown et al., 2017; Eykholt et al., 2018; Thys et al., 2019; Xu et al., 2020). Their diverse appearances and placement create a challenging defense problem: a system must distinguish adversarial content from benign visual structure, recognize unfamiliar attacks, and identify the regions responsible for its decisions. Practical patch defense therefore requires more than a binary attack prediction or an isolated segmentation mask. It must connect reliable detection with precise localization through interpretable visual evidence.

Existing defenses address different aspects of this problem, including robust classification, patch localization, and image restoration (Xiang et al., 2022; Liu et al., 2022a; Jing et al., 2024; Wei et al., 2025). However, these capabilities alone do not explain how evidence at a particular location supports an global-level attack decision. Localization without attack-level discrimination can mistake benign structures for adversarial regions, while detection without spatial evidence offers limited guidance for inspection or intervention. Figure 1 illustrates this gap: DIFFender (Wei et al., 2025) predicts spurious patch regions on a clean image, whereas the proposed PatchTrace connects its detection decision to prototype evidence and spatial localization. The accompanying comparisons Figure 1 also highlight a broader challenge: strong general-purpose segmentation model SAM 3 (Carion et al., 2026) and vision–language LLaVa-NeXT (Liu et al., 2024) capabilities do not automatically translate into reliable adversarial patch understanding. We examine vision–language reasoning by following the SmuDPatch (Kumar et al., 2026) protocol, we evaluate LLaVA-NeXT on 20,000 patched images from each benchmark (40,000 in total) using the prompt, "Is this image attacked by any adversarial patch?" LLaVA-NeXT identifies 56.6% of SmuDPatch images and 90.3% of ImageNet-Patch images as attacked, revealing uneven sensitivity across attack datasets.

Interpretability provides a natural bridge between deciding whether an attack exists and identifying where it occurs. Although patches occupy visible image regions, their appearance can resemble natural textures (Hu et al., 2021), making visual saliency alone an insufficient indicator of adversarial content. A useful defense must expose the evidence that distinguishes suspicious patterns from normal image content (Hofman et al., 2024; Kumar & Agarwal, 2025). Prototype-based models support this reasoning by relating predictions to representative visual concepts (Nauta et al., 2023). AdvPatchXAI (Kumar & Agarwal, 2025) applies this principle to global-level patch detection, but the broader challenge remains: how a defense can use a shared, interpretable representation to support both global detection and dense localization across diverse attacks?

We introduce **PatchTrace**, a prototype-based framework that unifies global-level attack detection (clean vs. patch) image and pixel-level patch localization (segmentation mask). PatchTrace learns separate clean and anomaly prototype banks, allowing the model to weigh evidence for normal content against evidence for adversarial patterns. A within-bank decorrelation objective encourages prototype diversity, while Cross-Prototype Attention Fusion (CPAF) combines information from the two banks for attack detection. This design makes the distinction between clean and adversarial evidence explicit and provides a common basis for detection and localization. The detector also gates the localization branch, so the framework produces a patch mask only when its detection decision indicates an attack.

To connect these global decisions with spatial evidence, we introduce the **Prototype Residual Attention Map (PRAM)**. PRAM emphasizes locations where anomaly evidence exceeds clean evidence and supplies a dense signal for patch localization. Instead of generating an independent post hoc explanation, PatchTrace constructs this signal from the prototype responses that underpin detection. The resulting global-to-local pipeline enables users to inspect both the visual concepts that support an attack prediction and the image regions that express those concepts. PatchTrace

thus treats interpretability as part of the defense mechanism and links its detection and localization outputs through shared evidence.

We evaluate PatchTrace across unseen patch patterns, different patch attack-generation methods, additional image noise, and physical-world datasets. Comparisons with specialized defenses and SAM 3 demonstrate strong detection and localization performance: PatchTrace achieves mean detection accuracies of 99.1% on SmuDPatch (Kumar et al., 2026) and 99.8% on ImageNet-Patch, alongside localization mIoU scores of 70.6% and 84.0%, respectively. Further evaluations on GAP (Wu et al., 2024) and three physical-world datasets (Pintor et al., 2023; Braunegg et al., 2020; Kumar et al., 2026) broaden our assessment of attack diversity and real-world transfer.

2 Related Work

We review adversarial patch defenses with an emphasis on how they identify attacks, localize suspicious regions, and explain their decisions.

Patch detection and localization. AdvPatchXAI (Kumar & Agarwal, 2025) and (Kumar & Agarwal, 2026) combines global-level attack detection with prototype-based explanations and evaluates generalization to unseen patches. Its prototype evidence supports classification, but it does not produce dense pixel-level patch masks. NAPGuard (Wu et al., 2024) targets naturalistic patches through critical feature modulation. SAC (Liu et al., 2022a) combines patch segmentation with robust mask shape completion and removal, while PAD (Jing et al., 2024) derives localization cues from semantic independence and spatial heterogeneity, then refines patch boundaries using SAM. DIFFender (Wei et al., 2025) integrates patch localization and image restoration through a text-guided diffusion model. These methods provide different mechanisms for recognizing or suppressing adversarial content; PatchTrace connects global-level recognition and dense localization within a shared prototype representation.

Selective intervention and explainability. Existing defenses already incorporate mechanisms to limit unnecessary image modification. SAC (Liu et al., 2022a) uses segmentation-based thresholding to determine whether patch removal is necessary, while PAD (Jing et al., 2024) applies adaptive localization thresholds and mask-overlap criteria. SATED (Jing et al., 2026) explicitly addresses false positives through semantic patch reasoning, providing decision explanations and text-guided inpainting. Thus, neither selective intervention nor explainability alone distinguishes PatchTrace. Figure 1 illustrates a false-positive example for DIFFender, rather than an inevitable failure of localization-based defenses.

To the best of our knowledge, PatchTrace is the first prototype-driven, unified end-to-end framework to combine interpretable global-level attack detection and, pixel-level patch localization. Its classifier weighs clean and anomaly prototype evidence to decide whether to activate localization. PRAM then uses the same prototype responses to highlight regions where anomaly evidence exceeds clean evidence, connecting why the model predicts an attack with where it locates the patch.

3 PatchTrace: Prototype-Guided Detection and Localization

PatchTrace jointly detects adversarial patches at the global level and localizes them at the pixel level. Figure 2 illustrates the architecture. We learn clean and anomaly prototypes to represent competing visual evidence, fuse their global responses for detection, and retain their spatial similarities for localization. This shared representation connects both predictions and exposes the prototype matches that inform the pipeline.

Problem setup. For an image $x \in \mathbb{R}^{3\times H\times W}$, we use an attack label $y \in \{0, 1\}$ and a binary patch mask $M \in \{0, 1\}^{H\times W}$, with $M = 0$ for clean images. PatchTrace predicts an attack probability $p(x)$ and a soft mask $\widetilde{M}(x)$; the detection decision controls the final binary mask $\widehat{M}(x)$.

3.1 Representing Clean and Adversarial Evidence

A patch can resemble benign image content, so a strong local response alone provides an incomplete attack cue. We introduce separate clean and anomaly banks to compare both types of evidence. A ConvNeXt-Tiny backbone Liu et al. (2022b) extracts fine and coarse features at strides 16 and 32.

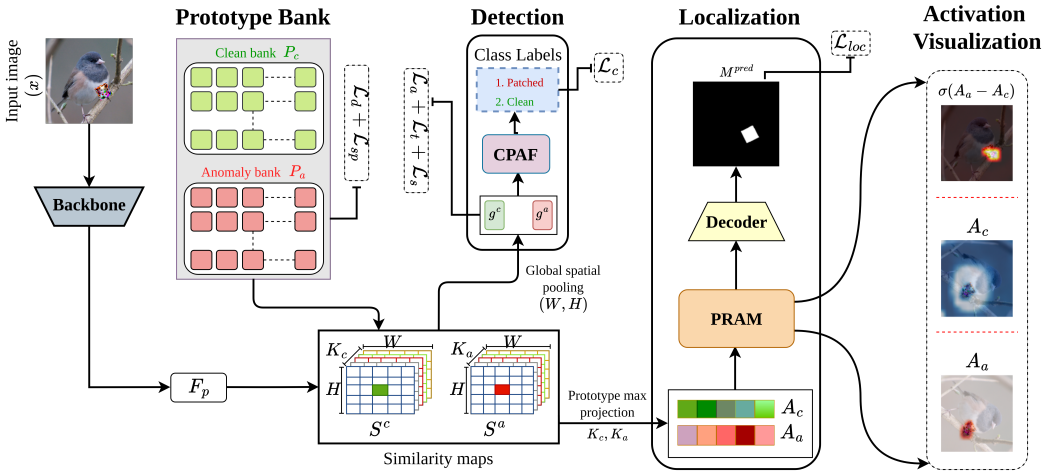


Figure 2: **Overview of the proposed PatchTrace.** A backbone extracts and projects image features to compare against separate clean (P_c) and anomaly (P_a) prototype banks. The resulting similarity maps (S^c, S^a) drive both tasks. For detection, we pool maps into evidence vectors (g^c, g^a), fused via CPAF, and classified under $\mathcal{L}_c$. If an attack is detected, the localization branch computes spatial evidence maps (A_c, A_a). The PRAM module then extracts residual anomaly evidence, guiding the decoder to predict the patch mask M^{pred} under $\mathcal{L}_{\text{loc}}$. **Visualizations (right):** Activation maps demonstrate how anomaly evidence overrides clean evidence around the patch region via $\sigma(A_a - A_c)$.

Channel projections produce F'_f and F'_c, and a further projection maps the fine features into a D_p-dimensional prototype space. For bank $b \in \{c, a\}$, we learn K_b prototypes and compute

$$s_k^b(u) = \langle \overline{f}_u, \overline{P}_k^b \rangle, \qquad g_k^b = \max_u s_k^b(u), \tag{1}$$

where u indexes spatial locations and the overbar denotes ℓ_2 normalization. The pooled response g_k^b captures the strongest match, while the similarity map S_k^b retains its spatial evidence. Mask-guided training assigns distinct roles to the two banks.

3.2 Detecting Attacks through Cross-Prototype Attention

Different patches can activate different combinations of responses. Cross-Prototype Attention Fusion (CPAF) therefore compares response patterns across the two banks. A bank-specific linear map embeds each scalar g_k^b into a D_h-dimensional token, and a learnable summary token precedes each sequence. At each layer, clean tokens attend to anomaly tokens, then anomaly tokens attend to the updated clean tokens. A fusion MLP combines the final summary tokens into z_{cpaf}.

To retain direct measures of response strength, we also provide the classifier with the mean, maximum, and mean difference of the bank responses:

$$\begin{aligned} d &= [\mu(g^c), \max g^c, \mu(g^a), \max g^a, \mu(g^a) - \mu(g^c)], \\ p(x) &= \sigma(\phi_{\text{cls}}([z_{\text{cpaf}} \,\|\, d])), \end{aligned} \tag{2}$$

Here μ denotes the mean, σ denotes sigmoid, and $\|$ denotes concatenation. CPAF models relationships between bank responses, while the indexed similarity maps retain individual prototype matches for inspection.

3.3 Localizing Patches through Residual Prototype Evidence

Global pooling summarizes attack evidence but removes its coordinates. The Prototype Residual Attention Map (PRAM) recovers a spatial cue from the same similarity maps by comparing the strongest anomaly and clean responses:

$$R_f(u) = \sigma\left(\max_k s_k^a(u) - \max_k s_k^c(u) \right). \tag{3}$$

Values above 0.5 indicate stronger anomaly evidence. We also average-pool each similarity channel to the coarse grid, repeat this comparison, and upsample the resulting map to obtain $R_c^{\uparrow}$. This second scale supplies a coarser view of prototype competition.

To refine patch boundaries, the decoder combines both PRAM maps with fine and coarse visual features:

$$\widetilde{M} = \sigma\big(\mathrm{Up}_{H,W}\big(\phi_{\mathrm{loc}}([R_f \,\|\, R_c^{\uparrow} \,\|\, F_f' \,\|\, F_c'^{\uparrow}])\big)\big)\,. \tag{4}$$

PRAM guides localization through relative prototype evidence, and the feature branches supply spatial detail. At inference, the detector suppresses the output mask when it predicts a clean image:

$$\widehat{M}(x) = \mathbf{1}[p(x) > \tau_{\mathrm{det}}]\,\mathbf{1}[\widetilde{M}(x) > \tau_{\mathrm{mask}}]. \tag{5}$$

The implementation applies this gate after decoding.

3.4 Joint Training and Loss Functions

We first warm up the backbone and fine-feature projection with the contrastive loss $\mathcal{L}_{\text{NT-Xent}}$ (Appendix C.1). We then jointly optimize detection and localization. Let i index a minibatch of N images, $p_i = p(x_i)$, and $\mathcal{I}_t = \{i : y_i = t\}$. Each objective below addresses a distinct requirement: classification predicts attack presence, alignment assigns spatial roles, prototype regularization structures the evidence, and mask supervision learns patch extent.

global-level classification ($\mathcal{L}_c$). Binary cross-entropy supplies clean-versus-patched supervision:

$$\mathcal{L}_c = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log p_i + (1-y_i)\log(1-p_i)\right]. \tag{6}$$

Spatial alignment ($\mathcal{L}_a^c, \mathcal{L}_a^a$). Classification alone does not constrain where prototypes respond. We therefore encourage clean prototypes to match non-patch content and anomaly prototypes to match patch regions. Let $m_i(u)$ denote the binary patch mask on the prototype grid. The masked batch maximum

$$q_k^b(W; v) = \max_{i,u}\left[W_i(u)s_{i,k}^b(u) + (1 - W_i(u))v\right] \tag{7}$$

selects a prototype's strongest match across images and locations, with fill value v outside the region W. Clean alignment uses

$$\mathcal{L}_a^c = -\frac{1}{K_c}\sum_{k=1}^{K_c} q_k^c(1-m;0). \tag{8}$$

Anomaly alignment rewards patch matches and penalizes non-patch responses above tolerance δ:

$$\mathcal{L}_a^a = -\frac{1}{K_a}\sum_{k=1}^{K_a} q_k^a(m; v_{\mathrm{inv}}) + \lambda_{\mathrm{out}}\left[\frac{1}{K_a}\sum_{k=1}^{K_a} q_k^a(1-m; v_{\mathrm{inv}}) - \delta\right]_+, \tag{9}$$

where $[t]_+ = \max(t, 0)$ and $v_{\mathrm{inv}} < -1$. We denote the weighted alignment component by

$$\mathcal{L}_a = \lambda_a^c \mathcal{L}_a^c + \lambda_a^a \mathcal{L}_a^a. \tag{10}$$

Separate coefficients preserve the different alignment strengths that the two banks use.

Within-bank decorrelation ($\mathcal{L}_d$). Several prototypes can otherwise converge toward similar directions. For $C_{rs}^b = \langle \overline{P}_r^b, \overline{P}_s^b \rangle$, we discourage redundancy within each bank:

$$\mathcal{L}_d^b = \frac{\sum_{r \neq s}[C_{rs}^b - \gamma_{\mathrm{dec}}]_+^2}{K_b(K_b-1) + \epsilon_d}, \qquad \mathcal{L}_d = \mathcal{L}_d^c + \mathcal{L}_d^a. \tag{11}$$

This soft penalty promotes diversity without imposing orthogonality.

Cross-bank separation ($\mathcal{L}_s$). Within-bank diversity does not distinguish clean and anomaly directions. We therefore penalize each prototype's strongest match to the other bank:

$$\mathcal{L}_s^{b\to b'} = \frac{1}{K_b}\sum_r \Big[\max_s \langle \overline{P}_r^b, \overline{P}_s^{b'}\rangle + \gamma_{\text{sep}}\Big]_+, \quad b \neq b',$$
$$\mathcal{L}_s = \mathcal{L}_s^{c\to a} + \mathcal{L}_s^{a\to c}. \tag{12}$$

The positive margin encourages cross-bank similarities at or below $-\gamma_{\text{sep}}$.

Class-conditional activation contrast ($\mathcal{L}_{\text{ctr}}$). Prototype directions alone do not ensure different responses to clean and patched images. Let $\mu_b^t = (|\mathcal{I}_t|K_b)^{-1}\sum_{i\in\mathcal{I}_t}\sum_k g_{i,k}^b$. We encourage stronger anomaly responses on patched images and stronger clean responses on clean images:

$$\mathcal{L}_{\text{ctr}} = [\gamma_{\text{ctr}} - (\mu_a^1 - \mu_a^0)]_+ + [\gamma_{\text{ctr}} - (\mu_c^0 - \mu_c^1)]_+. \tag{13}$$

We set this loss to zero when the batch lacks either class.

Activation regularization ($\mathcal{L}_t, \mathcal{L}_{\text{sp}}$). We balance a preference for large response magnitudes with an ℓ_1 penalty:

$$\mathcal{L}_t = \sum_{b\in\{c,a\}} \frac{1}{NK_b}\sum_{i=1}^N\sum_{k=1}^{K_b}\left[1-\tanh^2(g_{i,k}^b)\right], \tag{14}$$

$$\mathcal{L}_{\text{sp}} = \sum_{b\in\{c,a\}} \frac{1}{NK_b}\sum_{i=1}^N\sum_{k=1}^{K_b}|g_{i,k}^b|. \tag{15}$$

Both losses sum separate bank means. Because cosine responses lie in $[-1,1]$, $\mathcal{L}_t$ favors large absolute responses but does not produce binary activations.

Dense mask supervision ($\mathcal{L}_{\text{loc}}$). Alignment selects strong local matches; dense supervision teaches the decoder the full patch extent. For the ground-truth patched subset $\mathcal{I}_1$, pixelwise BCE supervises individual pixels and Dice loss supervises overlap:

$$\mathcal{L}_{\text{mask-bce}} = -\frac{1}{|\mathcal{I}_1|HW}\sum_{i\in\mathcal{I}_1}\sum_u\left[M_{i,u}\log\widetilde{M}_{i,u} + (1-M_{i,u})\log(1-\widetilde{M}_{i,u})\right], \tag{16}$$

$$\mathcal{L}_{\text{dice}} = \frac{1}{|\mathcal{I}_1|}\sum_{i\in\mathcal{I}_1}\left[1 - \frac{2\sum_u \widetilde{M}_{i,u}M_{i,u} + \epsilon}{\sum_u \widetilde{M}_{i,u} + \sum_u M_{i,u} + \epsilon}\right], \tag{17}$$

$$\mathcal{L}_{\text{loc}} = \lambda_{\text{mask-bce}}\mathcal{L}_{\text{mask-bce}} + \lambda_{\text{dice}}\mathcal{L}_{\text{dice}}. \tag{18}$$

Here u indexes input pixels. We set $\mathcal{L}_{\text{loc}} = 0$ when $\mathcal{I}_1$ is empty. Training selects examples by their true labels and propagates localization gradients through the shared features and PRAM, independently of detector predictions.

Complete objective. We combine all supervised terms as

$$\mathcal{L}_{\text{det}} = \lambda_c\mathcal{L}_c + \mathcal{L}_a + \lambda_d\mathcal{L}_d + \lambda_s\mathcal{L}_s + \lambda_{\text{ctr}}\mathcal{L}_{\text{ctr}} + \lambda_t\mathcal{L}_t + \lambda_{\text{sp}}\mathcal{L}_{\text{sp}}, \tag{19}$$
$$\mathcal{L} = \mathcal{L}_{\text{det}} + \mathcal{L}_{\text{loc}}. \tag{20}$$

The ablations isolate alignment $\mathcal{L}_a$, decorrelation $\mathcal{L}_d$, their joint contribution, and localization supervision $\mathcal{L}_{\text{loc}}$. Appendix C provides the warm-up objective, masking conventions, and numerical hyperparameters.

4 Results and Analysis

This section first introduces the benchmark datasets and evaluation protocol used to assess the proposed PatchTrace. We then analyze the results in two stages: global-level (clean vs. patch) adversarial patch detection and pixel-level patch localization. This structure allows us to evaluate both the global and local effectiveness of PatchTrace under unseen-patch and noisy settings.

Table 1: **Mean cross-patch detection accuracy (%) on COCO.** Each column P-i corresponds to a model that trains on ImageNet images with patch i. We evaluate this model separately on each of the other nine patch patterns using equal numbers of clean and patched COCO images, then average the nine accuracies. **Silent** introduces no additional noise; **Noisy** further averages results across Gaussian, shot, and impulse noise at severity 2. **Mean** averages the ten columns. Bold marks the best method for each setting and column.

Method	Setting	P-0	P-1	P-2	P-3	P-4	P-5	P-6	P-7	P-8	P-9	Mean
AdvPatchXAI (Kumar & Agarwal, 2025)	Silent	90.0	85.3	86.1	75.9	96.2	82.4	78.6	87.1	92.7	95.2	87.0
	Noisy	79.1	79.8	74.6	64.6	86.5	59.7	72.7	71.0	76.4	84.0	74.8
(Kumar & Agarwal, 2026)	Silent	73.1	86.0	78.3	81.2	85.6	80.4	79.2	78.1	76.2	81.4	79.9
	Noisy	60.1	77.3	71.2	75.2	80.3	75.4	73.3	73.2	74.4	79.0	73.9
PatchTrace (ours)	Silent	**100.0**	**100.0**	**99.9**	**99.3**	**100.0**	**100.0**	**100.0**	**99.9**	**99.5**	**99.0**	**99.8**
	Noisy	**97.3**	**99.3**	**97.4**	**90.8**	**99.0**	**98.2**	**98.5**	**95.1**	**94.9**	**89.8**	**96.0**

4.1 Datasets and Evaluation Protocol

We train PatchTrace exclusively on the ImageNet training split of the (Kumar & Agarwal, 2026; 2025) benchmark and evaluate on the COCO test splits of this benchmark and OOD-SmuDPatch (Kumar et al., 2026). We use ten patch patterns from each benchmark. For each training run, we use clean images and images containing one source patch, then evaluate on the remaining nine source-benchmark patches and all ten SmuDPatch patches without retraining or adaptation, all the patches are evaluated individually with an equal number of patched and clean images in the val set, and the average performance is reported. These protocols assess cross-image-dataset transfer to unseen patch patterns and further generalization to a different attack-generation method, respectively. To examine broader attack diversity, we evaluate on GAP (Wu et al., 2024), which combines INRIA (Dalal & Triggs, 2005) and MS COCO (Lin et al., 2014) images with 25 patch types from eight attack methods, including 15 naturalistic and ten non-naturalistic patches. Physical-world evaluations on ImageNet-Patch (Pintor et al., 2023), SmuDPatch (Kumar et al., 2026), and APRICOT (Braunegg et al., 2020) further test transfer beyond digitally applied patches.

Silent and Noisy settings. The *Silent* setting evaluates unseen patches without additional noise, while the *Noisy* setting evaluates unseen patches with noise types absent during training. Appendix D provides the full evaluation protocol.

4.2 Global-level Attack Detection

We evaluate whether PatchTrace correctly classifies each input image as clean or containing an adversarial patch (clean vs. patch), which we refer to as global-level attack detection. We assess generalization to unseen patch patterns and a different attack-generation method under both Silent and Noisy settings.

Unseen-patch generalization on ImageNet-Patch. Table 1 shows that PatchTrace outperforms both baselines across all ten splits in both settings. It achieves 99.8% mean accuracy in Silent evaluation and 96.0% in Noisy evaluation, exceeding the strongest baseline mean by 12.8 and 21.2 percentage points, respectively. Noise reduces PatchTrace's mean accuracy by only 3.8 points, compared with 12.2 points for AdvPatchXAI. These results show that PatchTrace learns transferable anomaly evidence rather than patch templates, and therefore generalizes reliably across unseen patch families and noisy inputs.

Cross-dataset generalization to SmuDPatch. Table 2 shows that PatchTrace generalizes to SmuDPatch without adaptation, achieving 99.1% mean accuracy in Silent evaluation and 84.0% in Noisy evaluation. These results exceed AdvPatchXAI by 12.6 and 2.2 percentage points, respectively. PatchTrace leads across all ten Silent splits and seven Noisy splits. Its lower accuracy on P-3, P-7, and P-9 under noise highlights remaining sensitivity to simultaneous shifts in attack generation and input corruption. We want to highlight that the patches proposed in SmuDPatch are not only developed using different generation approaches but also exhibit substantially different visual patterns and smooth, deformable patch boundaries. PatchTrace still detects them reliably, which shows that its prototypes capture patch-specific statistics rather than the shape or texture of any single attack.

Generalization across diverse attack methods. We further evaluate PatchTrace on GAP (Wu et al., 2024), which contains 3,649 patched images from INRIA (Dalal & Triggs, 2005) and MS COCO

Table 2: **Mean detection accuracy (%) on SmuDPatch.** Each column P-i corresponds to a model that trains on ImageNet images with source patch i. We evaluate it separately on all ten SmuDPatch patterns using equal numbers of clean and patched COCO images, then average the ten accuracies without retraining or adaptation.

Method	Setting	P-0	P-1	P-2	P-3	P-4	P-5	P-6	P-7	P-8	P-9	Mean
AdvPatchXAI (Kumar & Agarwal, 2025)	Silent	88.9	95.9	85.4	82.7	95.8	68.7	89.9	89.7	75.8	91.8	86.5
	Noisy	84.5	93.2	80.4	**80.8**	92.0	66.3	80.3	**82.7**	66.7	**90.8**	81.8
PatchTrace (ours)	Silent	**99.9**	**99.9**	**99.8**	**97.9**	**99.9**	**99.9**	**100.0**	**98.8**	**99.5**	**95.3**	**99.1**
	Noisy	**88.3**	**94.8**	**90.0**	59.3	**94.4**	**90.2**	**94.8**	76.2	**88.2**	63.4	**84.0**

Table 3: **Cross-patch localization mIoU (%) on COCO.** For methods that require training, each P-i column corresponds to ImageNet training on patch i and averages COCO localization results across the other nine patches patterns.

Method	Setting	P-0	P-1	P-2	P-3	P-4	P-5	P-6	P-7	P-8	P-9	Mean
NAPGuard (Wu et al., 2024)	Silent	56.6	58.7	54.2	36.3	61.8	56.5	48.1	37.8	49.9	46.3	50.6
	Noisy	27.8	45.4	38.7	26.7	46.5	51.0	36.9	28.9	35.7	26.2	36.4
DIFFender (Wei et al., 2025)	Silent	26.8	30.3	23.0	24.6	24.7	29.5	25.9	24.1	25.7	29.6	26.4
	Noisy	18.1	22.1	18.4	16.2	19.7	17.7	17.3	21.3	24.1	22.3	19.7
SAM 3 (Carion et al., 2026)	Silent	18.2	56.8	37.6	58.8	36.4	60.1	61.6	19.9	19.0	26.3	39.5
	Noisy	2.0	23.9	10.1	22.1	14.7	31.5	41.6	8.3	7.1	9.3	17.1
PatchTrace (ours)	Silent	**88.0**	**90.4**	**86.1**	**75.8**	**90.1**	**84.3**	**87.8**	**81.5**	**81.4**	**74.8**	**84.0**
	Noisy	**72.8**	**80.2**	**69.1**	**39.6**	**80.8**	**65.8**	**73.4**	**57.4**	**57.9**	**47.4**	**64.4**

(Lin et al., 2014), and spans 25 distinct patch types from eight attack methods, including 15 naturalistic and 10 non-naturalistic patches. We balance this evaluation with 3,649 clean images from INRIA and the MS COCO validation set. This diversity tests generalization across patch appearances and generation methods. PatchTrace achieves 59.8% detection accuracy, exceeding AdvPatchXAI's 45.95% by 13.85 percentage points. Although PatchTrace improves over AdvPatchXAI, its modest accuracy highlights the challenge of generalizing across this broader range of attacks.

4.3 Pixel-Level Patch Localization

We evaluate how accurately PatchTrace identifies the pixels that belong to an adversarial patch. We measure overlap between predicted and ground-truth patch masks using mean intersection-over-union (mIoU) under Silent and Noisy settings.

Unseen-patch generalization on ImageNet-Patch. Table 3 shows that PatchTrace achieves 84.0% and 64.4% mean mIoU in Silent and Noisy evaluation, respectively, outperforming all tabulated baselines across all ten splits in both settings. PatchTrace exceeds NAPGuard, the strongest tabulated baseline, by 33.4 and 28.0 percentage points. SAM 3 reaches 39.5% and 17.1%, highlighting the challenge that unseen adversarial patches pose for general-purpose segmentation. Despite these gains, noise reduces PatchTrace's mIoU by 19.6 points, indicating a remaining challenge for precise localization. We also evaluate PAD (Jing et al., 2024); PatchTrace surpasses its mean mIoU by 38.5 and 1.5 points in Silent and Noisy evaluation, respectively.

Cross-dataset generalization to SmuDPatch. Table 4 shows that PatchTrace achieves 70.6% mIoU in Silent evaluation and 34.8% in Noisy evaluation, exceeding DIFFender by 46.7 and 18.5 percentage points, respectively. PatchTrace leads across all ten Silent columns and eight Noisy columns, while SAM 3 achieves only 9.4% and 0.5% mean mIoU. These results support localization transfer to SmuDPatch without adaptation. However, PatchTrace's 35.8-point decline under noise highlights the difficulty of localizing unfamiliar deformable patches under corruption.

4.4 Physical-World Evaluation

To assess whether PatchTrace detects adversarial patches in real-world imagery, we evaluate PatchTrace on physical-world images from ImageNet-Patch (Pintor et al., 2023), SmuDPatch (Kumar et al., 2026), and APRICOT (Braunegg et al., 2020), containing 90, 152, and 873 patched images, respectively. Since these evaluation sets contain only patched images, so we report recall, the propor-

Table 4: **Patch localization mIoU (%) on SmuDPatch.** We evaluate on COCO images without training or adapting the methods to SmuDPatch. **Silent** introduces no additional noise; **Noisy** averages results across Gaussian, shot, and impulse noise at severity 2. **Mean** averages the ten columns. Bold marks the best method for each setting and column.

Method	Setting	P-0	P-1	P-2	P-3	P-4	P-5	P-6	P-7	P-8	P-9	Mean
DIFFender (Wei et al., 2025)	Silent	24.4	24.1	23.5	23.6	24.5	24.1	23.9	21.7	23.9	25.6	23.9
	Noisy	18.7	14.6	16.8	**15.4**	17.8	13.2	15.1	16.8	18.8	**15.9**	16.3
SAM 3 (Carion et al., 2026)	Silent	8.8	1.8	35.6	5.9	18.4	9.5	13.0	0.0	0.8	0.0	9.4
	Noisy	0.1	0.0	3.6	0.1	1.3	0.1	0.3	0.0	0.0	0.0	0.5
PatchTrace (ours)	Silent	**79.7**	**82.7**	**76.4**	**58.1**	**78.5**	**77.1**	**83.0**	**55.1**	**66.5**	**49.3**	**70.6**
	Noisy	**38.7**	**56.9**	**42.2**	14.9	**48.4**	**34.8**	**53.1**	**19.0**	**28.0**	12.3	**34.8**

Table 5: Loss ablation on the nine held-out COCO patch types after training on Patch 1. Each column after Full names the loss that the variant omits. Entries report mean scores (%).

Metric	Full	$\mathcal{L}_d$	$\mathcal{L}_a$	$\mathcal{L}_d + \mathcal{L}_a$	$\mathcal{L}_s$	$\mathcal{L}_{\text{ctr}}$	$\mathcal{L}_t$	$\mathcal{L}_{\text{sp}}$	$\mathcal{L}_{\text{loc}}$
Detection F1 (%)	**99.9**	99.6	99.5	96.1	99.9	99.9	99.9	99.9	99.8
Localization mIoU (%)	**90.4**	74.5	81.8	86.0	88.5	89.6	86.3	85.9	0.9

tion of patched images that the detector correctly identifies as attacks. PatchTrace achieves 95.6% recall on ImageNet-Patch and 84.9% on SmuDPatch, exceeding AdvPatchXAI by 6.5 and 34.3 percentage points, respectively. On APRICOT, PatchTrace increases recall from 4.3% to 23.7%, a gain of 19.4 points. These improvements demonstrate stronger attack recognition across all three datasets, while the low absolute recall on APRICOT highlights a substantial remaining challenge for physical-world generalization. Since this evaluation contains no clean images, it measures attack sensitivity without assessing false-positive behavior.

4.5 Ablation of Loss Functions

To further analyze the loss design of PatchTrace, Table 5 evaluates loss ablations after training on Patch 1 and testing on the nine unseen COCO patch types. The full objective reaches 99.99% detection F1 and 90.43% localization mIoU. Removing localization supervision reduces mIoU to 0.92%, although detection F1 remains 99.80%; the prototype detector therefore identifies patch presence without learning an accurate dense mask. Removing decorrelation or alignment reduces mIoU to 74.59% or 81.83%, respectively. Removing both lowers detection F1 to 96.16%, but its 86.03% mIoU exceeds either individual ablation, so the two losses do not contribute additively. The remaining regularizers have smaller effects on detection, while their removal still lowers localization mIoU.

Computational Cost PatchTrace processes a 224×224 image in 8.95 ± 0.05 ms with 33.4M parameters and 0.56 GB peak GPU memory on a single NVIDIA H200 GPU. NAPGuard (Wu et al., 2024) runs faster at 4.53 ± 0.07 ms with 7.0M parameters, but it delivers much weaker localization performance (50.6/36.4 Silent/Noisy mIoU) than PatchTrace (84.0/64.4). In contrast, PAD (Jing et al., 2024) and DIFFender (Wei et al., 2025) incur much higher cost: PAD requires 5369.63 ms per image, and DIFFender requires 120.68 ms with 1066.2M parameters. Overall, PatchTrace offers the best accuracy–efficiency trade-off, combining strong cross-patch localization with far lower cost than SAM- and diffusion-based defenses. Appendix H provides full computational details.

5 Conclusion

We introduce PatchTrace, a unified prototype-driven framework for interpretable adversarial patch detection and localization. Dual clean and anomaly prototype banks support global-level decisions, while PRAM connects their spatial responses to pixel-level localization. Experiments across unseen patch patterns, attack-generation methods, noisy inputs, and physical-world datasets demonstrate stronger detection and localization than competing baselines. The low recall on APRICOT highlights remaining challenges. These findings support shared prototype evidence as a basis for accurate and inspectable adversarial patch defense.

AI USE STATEMENT

We use ChatGPT to polish the writing and improve the clarity and readability of the manuscript.

ETHICS STATEMENT

PatchTrace aims to improve the security and interpretability of vision systems through adversarial patch detection and localization. Our evaluation uses existing image benchmarks and adversarial patch datasets. Although this work focuses on defense, insights into detector behavior may also help adversaries develop stronger attacks. Our results do not establish deployment safety: severe corruption and challenging physical-world examples remain important failure cases. We therefore emphasize transparent reporting of these limitations and application-specific validation before deployment in safety-critical settings.

REPRODUCIBILITY STATEMENT

Section 3 describes PatchTrace's prototype banks, detection and localization modules, and training objectives. Appendices B and C provide the detailed architecture and loss formulations. Appendix C.2 specifies initialization, augmentations, optimization schedules, hyperparameters, and checkpoint selection. Appendix D documents dataset subsets, training and evaluation protocols, corruption settings, metrics, and score aggregation. Together, these details specify the experimental procedure for reproducing our results.

REFERENCES

Anneliese Braunegg, Amartya Chakraborty, Michael Krumdick, Nicole Lape, Sara Leary, Keith Manville, Elizabeth Merkhofer, Laura Strickhart, and Matthew Walmer. Apricot: A dataset of physical adversarial attacks on object detection. In *European Conference on Computer Vision*, pp. 35–50. Springer, 2020.

Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pp. 138846–138923, 2026.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PmLR, 2020.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pp. 886–893. Ieee, 2005.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 1625–1634. IEEE, 2018.

Omer Hofman, Amit Giloni, Yarin Hayun, Ikuya Morikawa, Toshiya Shimizu, Yuval Elovici, and Asaf Shabtai. X-detect: Explainable adversarial patch detection for object detectors in retail. *Machine Learning*, 113(9):6273–6292, 2024.

Yu-Chih-Tuan Hu, Bo-Han Kung, Daniel Stanley Tan, Jun-Cheng Chen, Kai-Lung Hua, and Wen-Huang Cheng. Naturalistic physical adversarial patch for object detectors. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7848–7857, 2021.

Lihua Jing, Rui Wang, Wenqi Ren, Xin Dong, and Cong Zou. Pad: Patch-agnostic defense against adversarial patch attacks. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24472–24481. IEEE, 2024.

Lihua Jing, Rui Wang, Jinwen Zhong, Runbo Li, and Zixuan Zhu. False positives matter: Multi-dimensional localization evaluation and training-free explainable adversarial patch defense. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 5557–5565, 2026.

Vishesh Kumar and Akshay Agarwal. A unified, resilient, and explainable adversarial patch detector. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 30387–30397. IEEE, 2025.

Vishesh Kumar and Akshay Agarwal. The unseen adversaries: Robust and generalized defense against adversarial patches. In Emtiyaz Khan, Yingzhen Li, Arno Solin, and Aaditya Ramdas (eds.), *Proceedings of The 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pp. 4528–4536. PMLR, 02–05 May 2026. URL https://proceedings.mlr.press/v300/kumar26a.html.

Vishesh Kumar, Guha Balakrishnan, and Akshay Agarwal. Breaking rigidity in adversarial patch attacks. In *European Conference on Computer Vision*, pp. 62–80. Springer, 2026.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llavanext: Improved reasoning, ocr, and world knowledge, 2024.

Jiang Liu, Alexander Levine, Chun Pong Lau, Rama Chellappa, and Soheil Feizi. Segment and complete: Defending object detectors against adversarial patch attacks with robust patch detection. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962. IEEE, 2022a.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11966–11976. IEEE, 2022b.

Meike Nauta, Jörg Schlötterer, Maurice Van Keulen, and Christin Seifert. Pip-net: Patch-based intuitive prototypes for interpretable image classification. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2744–2753. IEEE, 2023.

Maura Pintor, Daniele Angioni, Angelo Sotgiu, Luca Demetrio, Ambra Demontis, Battista Biggio, and Fabio Roli. Imagenet-patch: A dataset for benchmarking machine learning robustness against adversarial patches. *Pattern Recognition*, 134:109064, 2023.

Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 49–55. IEEE, 2019.

Xingxing Wei, Caixin Kang, Yinpeng Dong, Zhengyi Wang, Shouwei Ruan, Yubo Chen, and Hang Su. Real-world adversarial defense against patch attacks based on diffusion model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Siyang Wu, Jiakai Wang, Jiejie Zhao, Yazhe Wang, and Xianglong Liu. Napguard: Towards detecting naturalistic adversarial patches. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24367–24376. IEEE, 2024.

Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. {PatchCleanser}: Certifiably robust defense against adversarial patches for any image classifier. In *31st USENIX security symposium (USENIX Security 22)*, pp. 2065–2082, 2022.

Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world. In *European conference on computer vision*, pp. 665–681. Springer, 2020.

A Appendix

B Architecture Details

Features and prototypes. For a 224×224 input, ConvNeXt-Tiny supplies $384 \times 14 \times 14$ and $768 \times 7 \times 7$ features from zero-based stages 2 and 3. Two 1×1 Conv–BN–GELU blocks project them to D channels; a 1×1 Conv–BN block maps the fine features to D_p channels. These convolutions omit biases. We initialize prototypes with Kaiming-normal initialization and normalize vectors as $\overline{v} = v/\max(\|v\|_2, 10^{-6})$.

CPAF. Each bank shares a linear embedding $\psi_b : \mathbb{R} \to \mathbb{R}^{D_h}$ and prepends a summary token, giving $T^{b,(0)} = [t^b_{\text{cls}}; \psi_b(g^b_1); \ldots; \psi_b(g^b_{K_b})]$. Each layer updates the banks sequentially:

$$\begin{aligned} T^{c,(\ell)} &= \mathcal{B}^{(\ell)}_c(T^{c,(\ell-1)}, T^{a,(\ell-1)}, T^{a,(\ell-1)}), \\ T^{a,(\ell)} &= \mathcal{B}^{(\ell)}_a(T^{a,(\ell-1)}, T^{c,(\ell)}, T^{c,(\ell)}), \end{aligned} \tag{21}$$

where $\mathcal{B}(Q, K, V)$ denotes a cross-attention block. CPAF uses L layers, each with n_h heads, residual connections, post-LayerNorm, and a $D_h \to 2D_h \to D_h$ feed-forward network. The fusion and classification MLPs follow $2D_h \to D_h \to D_h$ and $(D_h + 5) \to 128 \to 1$, respectively. They use GELU and dropout p_{drop}. CPAF adds no prototype identity embeddings, so it compares response magnitudes within and across banks.

PRAM and decoder. Adaptive average pooling maps each similarity channel to 7×7 before the coarse PRAM takes the bankwise maxima. The decoder receives $2D + 2$ channels at 14×14. Three stages each perform $2\times$ bilinear upsampling followed by two 3×3 Conv–BN–GELU blocks with padding 1 and no convolutional bias. Their widths follow $(128, 64, 32)$. A final 1×1 convolution produces logits at 112×112; bilinear interpolation maps them to 224×224 before sigmoid. All bilinear feature and logit resizes use `align_corners=False`.

C Loss Conventions and Training Settings

C.1 Contrastive Warm-Up

We generate two augmented views of each training image and spatially average the projected fine features. A $768 \to 768 \to 128$ GELU projection head maps each view to a normalized embedding z_i. Let $\pi(i)$ index the other view of the same image among the $2N$ embeddings. The symmetric NT-Xent objective is

$$\mathcal{L}_{\text{NT-Xent}} = -\frac{1}{2N} \sum_{i=1}^{2N} \log \frac{\exp(z_i^\top z_{\pi(i)}/\tau_{\text{ssl}})}{\sum_{j \neq i} \exp(z_i^\top z_j/\tau_{\text{ssl}})}. \tag{22}$$

Warm-up optimizes the trainable backbone, fine-feature projection, and contrastive head through this loss. Joint training then optimizes the supervised objective in Equation equation 20.

Masking and reductions. Nearest-neighbor interpolation maps ground-truth masks to the prototype grid. Equations equation 8–equation 9 use zero masking for clean alignment and $v_{\text{inv}} = -10$ for anomaly alignment. If no patch survives on the grid, the inside term contributes the constant $-v_{\text{inv}}$ with zero gradient; outside suppression still applies. The implementation evaluates both BCE terms from logits for numerical stability.

C.2 Implementation and Optimization

Initialization and augmentation. We initialize ConvNeXt-Tiny with `timm`'s pretrained weights, freeze zero-based stages 0 and 1, and train the stem and later stages. We resize RGB inputs to 224×224 and apply ImageNet normalization. The loader jointly flips images and masks horizontally with probability 0.5 and applies image-only color jitter $(0.2, 0.2, 0.1, 0.02)$. For contrastive warm-up, we generate two independent views using random resized crops with scale $(0.2, 1.0)$, horizontal flips,

color jitter $(0.4, 0.4, 0.4, 0.1)$ with probability 0.8, grayscale conversion with probability 0.2, and Gaussian blur with kernel size 23 and $\sigma \in (0.1, 2.0)$. These SimCLR-style transformations (Chen et al., 2020) do not enforce patch retention.

Two-phase optimization. Phase A optimizes the trainable backbone parameters and contrastive head for up to E_{ssl} epochs. Phase B starts from the final warm-up state and jointly optimizes the backbone, projections, prototype banks, CPAF, classifier, and decoder for up to E_{joint} epochs. We use AdamW with separate parameter-group learning rates (Table 6). The smaller backbone learning rate limits changes to pretrained features, while larger rates allow the prototypes and task heads to adapt. Phase A uses cosine annealing. Phase B linearly increases detector learning rates from 0.1 times their base values over five epochs, then applies cosine annealing; the decoder uses cosine annealing throughout. We use weighted sampling with replacement to balance clean and patched examples in expectation, mixed precision, FP32 loss computation, gradient clipping at norm 5, and seed 42. We skip non-finite batches and run experiments on a single NVIDIA H200 GPU.

Early stopping and model selection. Phase A monitors contrastive training loss with patience 10. Phase B monitors

$$q_{\text{val}} = \text{F1}_{\text{val}} + 2\,\text{IoU}_{\text{val}} - 0.01\,\mathcal{L}_{\text{val}}, \tag{23}$$

with patience 15, using F1 and IoU on the $[0, 1]$ scale. Both phases require improvement above 10^{-4}. We select the supervised checkpoint with the highest qualifying validation score. Validation computes IoU from ungated predictions on patched images; testing applies the detection gate and assigns zero IoU to missed patches. We keep the hyperparameters fixed across training splits and change only the relevant loss terms for each ablation.

Table 6: PatchTrace hyperparameters. η_E, η_P, η_C, and η_D denote learning rates for the backbone and projections, prototype banks, CPAF and classifier, and decoder, respectively. E_{ssl} and E_{joint} denote epoch limits; ω_{wd} denotes weight decay.

Parameter	**Value**	**Parameter**	**Value**
(K_c, K_a)	$(256, 128)$	(D, D_p, D_h)	$(768, 256, 256)$
$(L, n_h, p_{\text{drop}})$	$(2, 4, 0.1)$	$(\tau_{\text{det}}, \tau_{\text{mask}})$	$(0.5, 0.5)$
λ_c	10	$(\lambda_a^c, \lambda_a^a)$	$(1, 2)$
λ_d	1	λ_s	0.3
λ_{ctr}	2	$(\lambda_t, \lambda_{\text{sp}})$	$(2, 0.1)$
$(\lambda_{\text{mask-bce}}, \lambda_{\text{dice}})$	$(1, 1)$	$(\lambda_{\text{out}}, \delta)$	$(1.5, 0.3)$
$(\gamma_{\text{dec}}, \gamma_{\text{sep}}, \gamma_{\text{ctr}})$	$(0.3, 0.3, 0.5)$	(ϵ_d, ϵ)	$(10^{-8}, 1)$
N	32	τ_{ssl}	0.5
(η_E, η_P)	$(10^{-4}, 3\times10^{-3})$	(η_C, η_D)	$(5\times10^{-3}, 10^{-3})$
η_{ssl}	10^{-4}	ω_{wd}	10^{-4}
$(E_{\text{ssl}}, E_{\text{joint}})$	$(30, 60)$		

D Datasets, Evaluation Protocols, and Metrics

D.1 Training Data and Cross-Patch Evaluation

The ImageNet-Patch benchmark (Kumar & Agarwal, 2026; 2025) contains clean and patched subsets from ImageNet and COCO (Lin et al., 2014). For each image dataset, it provides 2,000 clean images and 20,000 patched images covering ten ImageNet-derived patch patterns (Pintor et al., 2023). A 3:2 training–test split yields 1,200 training and 800 test images per clean or individual-patch subset.

We train exclusively on the ImageNet training split. Each run uses 1,200 clean images and 1,200 images containing one source patch. We repeat training for each of the ten source patches. We then

Table 7: **Training and cross-patch evaluation protocol.** Evaluation counts refer to one target patch. We reuse the clean subset across target patches and compute detection metrics separately for each balanced clean–patched pair. Noisy counts include all three corruption types at severity 2.

Stage	Benchmark	Images	Clean	Patched	Patch patterns
Training	ImageNet-Patch	ImageNet	1,200	1,200	One per run
Silent	ImageNet-Patch	COCO	800	800	Nine unseen
Silent	SmuDPatch	COCO	800	800	Ten unseen
Noisy	ImageNet-Patch	COCO	2,400	2,400	Nine unseen
Noisy	SmuDPatch	COCO	2,400	2,400	Ten unseen

evaluate each model on COCO test images containing the remaining nine patch patterns, without adaptation. Both the image dataset and patch pattern therefore change between training and testing, although the patches share the same generation approach.

For each unseen patch, we construct a balanced detection test set containing 800 clean and 800 patched COCO images. We evaluate the nine target patches separately and average their scores. We reuse the same clean test subset across these evaluations; we do not pool all patched subsets into a single imbalanced detection test set.

D.2 Cross-Dataset Evaluation on SmuDPatch

SmuDPatch (Kumar et al., 2026) applies twenty adversarial patches to 2,000 COCO images, producing 40,000 patched images, and provides a separate subset of 2,000 clean images. We select ten deformable patch patterns and use only their test partitions under the dataset's 3:2 split. This selection provides 800 clean images and 800 patched images per pattern.

We evaluate every ImageNet-trained model on all ten selected SmuDPatch patterns without retraining, fine-tuning, or adaptation. As above, each patch-specific detection evaluation pairs the 800 patched images with 800 clean images. Beyond the ImageNet-to-COCO image shift, SmuDPatch introduces different attack-generation approaches, visual patterns, and smooth, deformable boundaries.

D.3 Silent and Noisy Settings

The *Silent* setting evaluates unseen patches without additional corruption. The *Noisy* setting separately applies Gaussian, shot, and impulse noise at severity 2 to both clean and patched test images. Noise-only images remain negative examples, while patch-plus-noise images form positive examples. Training includes none of these noise types.

For each target patch and noise type, we evaluate 800 noisy clean images and 800 noisy patched images. We average scores across the three noise types and then across target patches. Thus, each patch-specific Noisy evaluation uses 2,400 corrupted clean and 2,400 corrupted patched images across the three corruptions. These images represent transformed versions of the original test images, not additional independent samples. Our severity ablation separately evaluates levels 1–5.

D.4 GAP and Physical-World Evaluation

GAP provides 3,649 patched images from INRIA and MS COCO, covering 25 patch types from eight attack methods. Its 15 naturalistic and ten non-naturalistic patches broaden the evaluation across patch appearances and generation methods. We construct a balanced detection set by adding 3,649 clean images: 288 from the INRIA test subset and 3,361 from MS COCO validation.

We also evaluate physical-world attack detection on ImageNet-Patch (Pintor et al., 2023), SmuDPatch (Kumar et al., 2026), and APRICOT (Braunegg et al., 2020), using 90, 152, and 873 patched

images, respectively. We use these datasets only for evaluation. Since the physical-world evaluation subsets contain only patched images, we report recall; these subsets cannot measure false-positive behavior.

D.5 Evaluation Metrics and Aggregation

Image-level detection. We treat attack detection as binary classification, with patched images as the positive class. Let TP, TN, FP, and FN denote the confusion-matrix counts. We compute

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \qquad \text{Precision} = \frac{TP}{TP + FP}, \tag{24}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \qquad \text{F1} = \frac{2TP}{2TP + FP + FN}. \tag{25}$$

We predict an attack when the detector probability exceeds 0.5. Because each patch-specific detection set contains equal numbers of clean and patched images, accuracy also equals balanced accuracy. We additionally compute AUROC and average precision from the continuous attack probabilities.

Pixel-level localization. We compute localization metrics on patched images with ground-truth masks. Let $\hat{y}_i$ denote the attack probability and $\hat{M}_i$ the soft mask. The final binary prediction applies both detection and mask thresholds:

$$B_i(u) = \mathbb{1}[\hat{y}_i > 0.5]\mathbb{1}[\hat{M}_i(u) > 0.5], \tag{26}$$

where u indexes pixels. For ground-truth mask M_i, we compute

$$\text{IoU}_i = \frac{\sum_u B_i(u) M_i(u)}{\sum_u \big(B_i(u) + M_i(u) - B_i(u) M_i(u)\big)}, \qquad \text{mIoU} = \frac{1}{N_p} \sum_{i=1}^{N_p} \text{IoU}_i, \tag{27}$$

where N_p denotes the number of patched images. A missed attack produces an empty prediction and therefore contributes zero IoU.

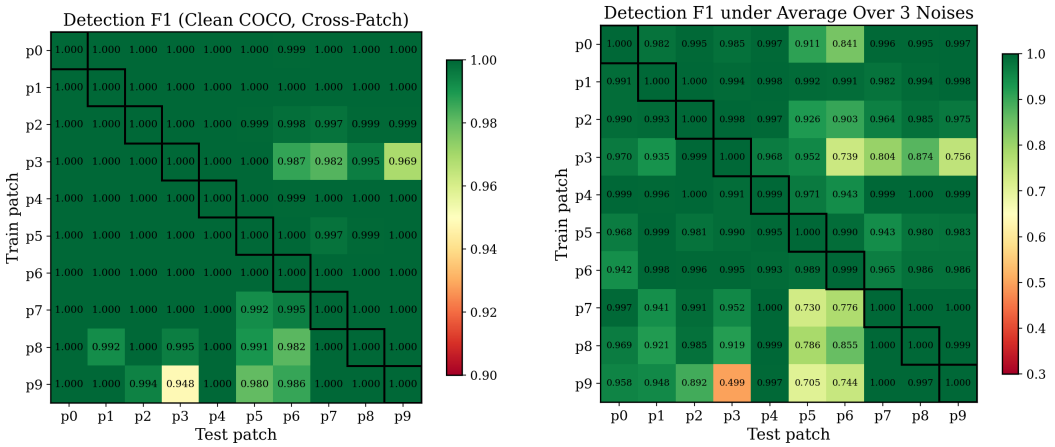
Aggregation across patches. For each source-patch model, we first compute each metric separately for every target patch. We then average over the nine unseen ImageNet-Patch patterns or all ten SmuDPatch patterns. Each P-i column reports this target-patch average for the model that trains on source patch i; the Mean column averages across the ten source-patch models. We express detection metrics and mIoU as percentages.

E Additional Results OOD Patch Detection and Localization

Figure 3 gives the full cross-patch generalization matrix for PatchTrace, where each row denotes the training patch and each column denotes the test patch. The top row reports detection F1, and the bottom row reports localization mean IoU, under both Silent and Noisy settings.

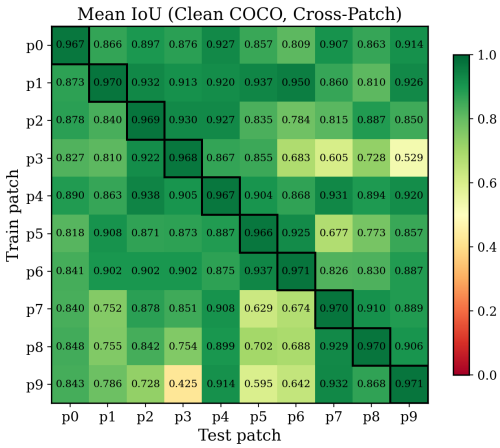
Under the **Silent** setting, PatchTrace shows near-perfect off-diagonal detection performance across almost all train–test patch pairs, with F1 values staying close to 1.0 even when the test patch differs from the training patch. This result confirms that the detector does not simply memorize the training patch family; instead, it learns transferable anomaly evidence that generalizes across patch styles. Localization also remains strong in the Silent setting, with most off-diagonal mean IoU values staying high, although a few train–test combinations show lower scores than detection. This gap is expected because localization requires finer spatial precision than global-level detection.

Under the **Noisy** setting, detection remains robust across most cross-patch pairs, but localization drops more noticeably. The heatmaps show that PatchTrace preserves high F1 across many off-diagonal entries even after averaging over Gaussian, shot, and impulse noise, which indicates strong robustness at the decision level. In contrast, the noisy localization matrix exhibits larger variation, and a few train–test combinations become substantially harder. In particular, patches with smoother or low-frequency appearance degrade more strongly under noise, because corruption distorts the spatial boundaries that the localizer must recover. Even so, PatchTrace retains useful localization

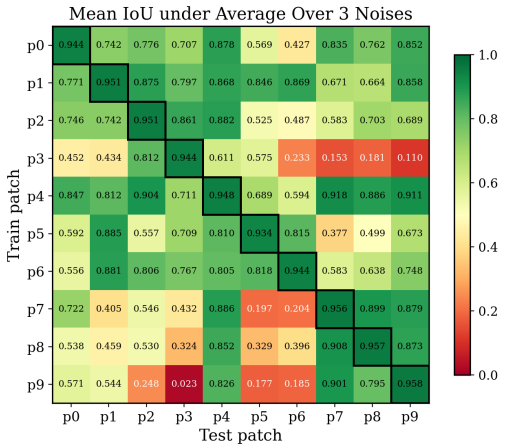


(a) Detection F1 under Silent setting.

(b) Detection F1 under Noisy setting.



(c) Localization mean IoU under Silent setting.



(d) Localization mean IoU under Noisy setting.

Figure 3: Cross-patch generalization heatmaps for PatchTrace. Rows denote the training patch and columns denote the test patch. The top row reports global-level detection F1, while the bottom row reports localization mean IoU. Silent denotes unseen-patch evaluation without noise, and Noisy denotes unseen-patch evaluation averaged over noise corruptions. The strong off-diagonal performance shows that PatchTrace generalizes beyond the patch used during training.

performance on most unseen patch pairs and maintains a clear separation between easier and harder transfer cases.

Overall, these heatmaps support the main claim of the paper: PatchTrace generalizes beyond the patch used during training. Detection transfers more consistently than localization, while noise mainly harms fine-grained mask prediction rather than global-level attack recognition. This behavior aligns with our main results and further shows that PatchTrace learns shared anomaly structure across patch families instead of relying on patch-specific templates.

E.1 Physical-World Localization Behavior

Figure 4 shows representative outputs of PatchTrace on images from the physical-world adversarial patch dataset (Pintor et al., 2023). For images that the detector classifies as patched, PatchTrace activates the localization branch and predicts a non-zero patch mask that aligns well with the adversarial region. For images that the detector classifies as *clean*, PatchTrace does not trigger the localization branch, leaves the input unchanged, and returns an all-zero predicted mask. This behavior is important in deployment because it avoids unnecessary dense processing on clean images and ensures that localization remains consistent with the global-level attack decision. The examples also show that

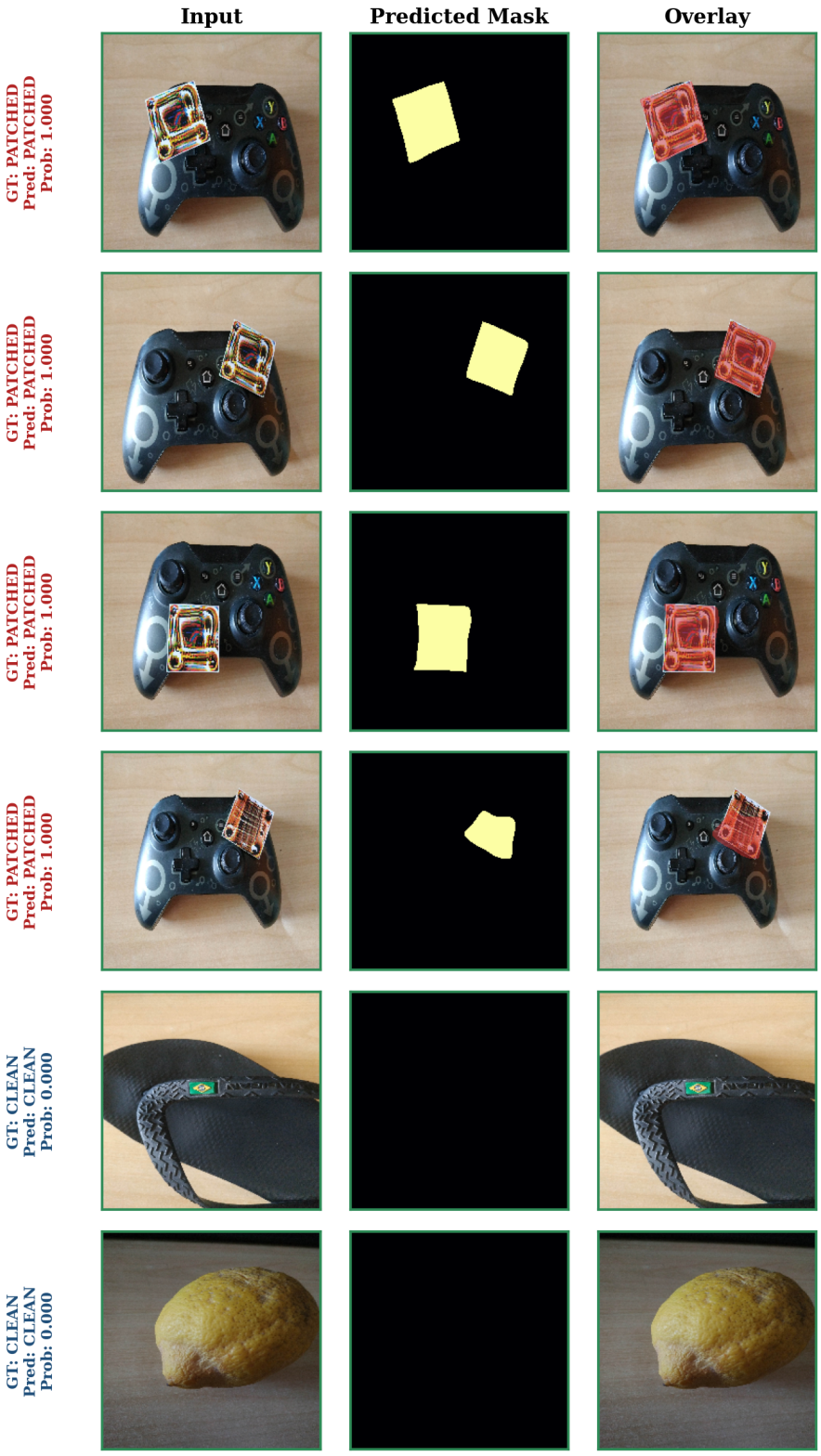
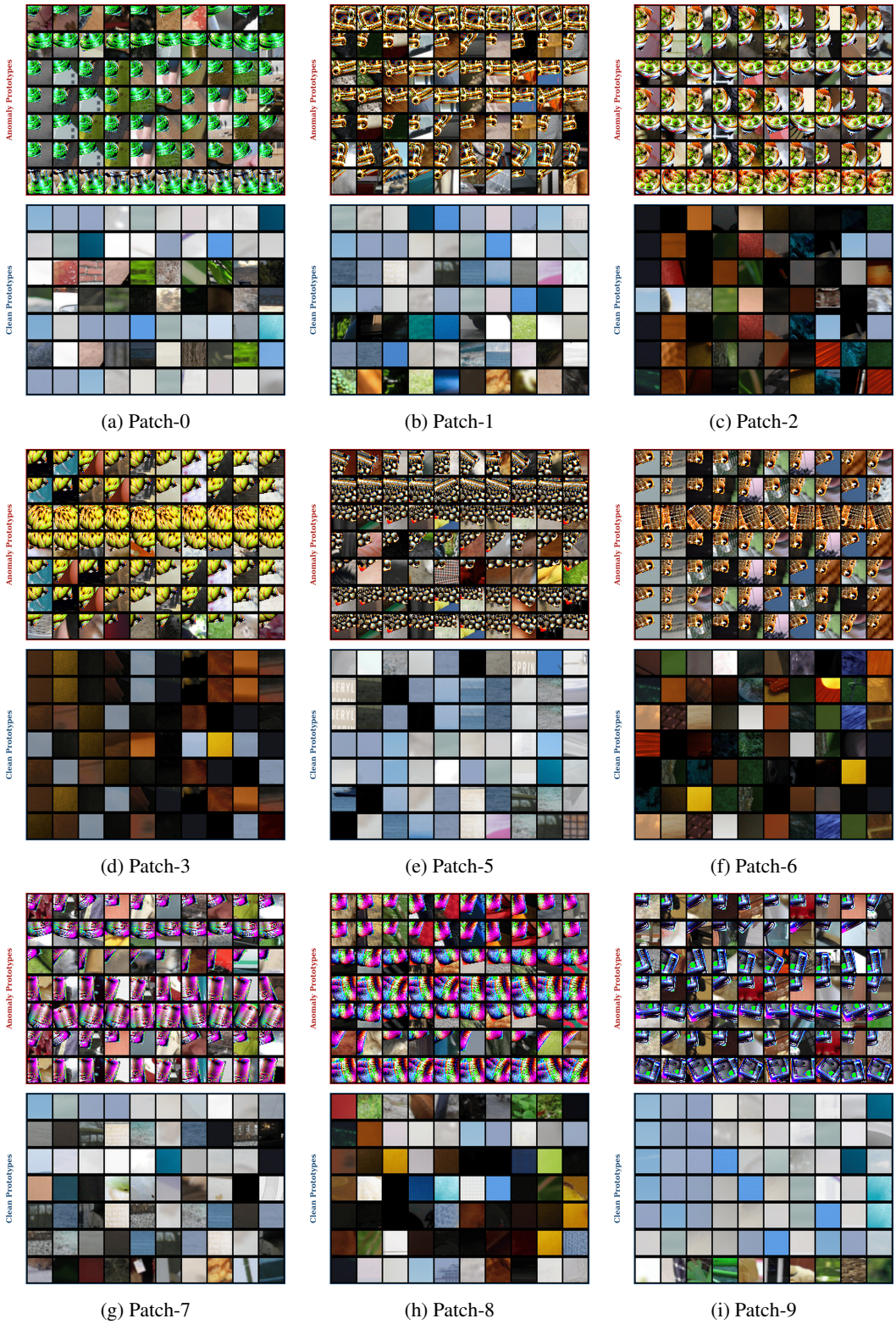


Figure 4: PatchTrace on physical-world images.

the predicted probability saturates near 1.0 for attacked inputs and near 0.0 for clean inputs, which further supports the intended global-to-local gating mechanism of PatchTrace.



(a) Patch-0 (b) Patch-1 (c) Patch-2

(d) Patch-3 (e) Patch-5 (f) Patch-6

(g) Patch-7 (h) Patch-8 (i) Patch-9

Figure 5: **Prototype visualization across patch types.** One per row visualized with their top-10 image patches learned during training for each patch.

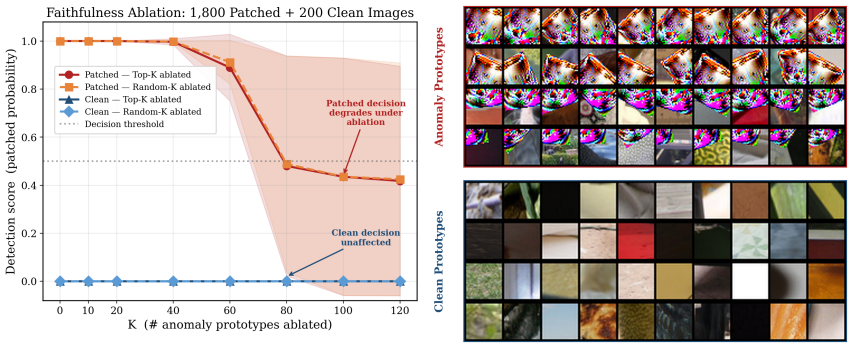


Figure 6: **Interpreting the anomaly prototype bank in PatchTrace. Left:** Faithfulness ablation of the anomaly prototype bank on patched and clean images. On patched inputs, the detection score degrades as anomaly prototypes are removed, while on clean inputs the prediction remains unchanged. This shows that the anomaly bank is necessary for patched-image detection but irrelevant for clean-image decisions. The similar behavior of top-K and random-K ablation at large K indicates a redundant distributed anomaly representation. The shaded regions show the mean $\pm$ standard deviation across the evaluated patched images for top-K and random-K anomaly-prototype ablation. **Right:** Example prototypes, one per row, visualized with their top-10 image patches when the proposed PatchTrace is trained on Patch-4.. Learned Prototypes of other adversarial patches visualized in the Appendix G.2

F Prototype-Based Explanation of PatchTrace

Accuracy alone does not establish interpretability. A prototype-based defense is useful only when prototype evidence directly drives the prediction. We therefore analyze the anomaly bank of PatchTrace in terms of decision relevance and cross-patch structure. We train PatchTrace on Patch-4, which gives the strongest overall transfer and the best localization under the hardest noisy setting, and test it on 9 OOD patch types with 1,800 patched images and 200 clean images. PatchTrace achieves 100% detection accuracy with zero false negatives and zero false positives, which shows that the anomaly bank does not memorize the training patch and instead learns a transferable representation of patchness.

Figure 6 (left) tests whether PatchTrace relies on this anomaly bank at inference time. We ablate K anomaly prototypes and measure the patched probability on both patched and clean images. On patched images, the score remains high under moderate ablation but drops sharply once we remove a large portion of the bank, falling from 0.997 at $K{=}40$ to 0.889 at $K{=}60$ and to 0.480 at $K{=}80$. On clean images, ablation leaves the prediction unchanged and keeps the patched probability near zero for both top-K and random-K ablations. At large K, top-K and random-K ablations behave almost identically (0.480 vs. 0.488 at $K{=}80$), which shows that PatchTrace does not rely on a few uniquely critical prototypes but instead distributes anomaly evidence across the bank.

Figure 6 (right) visualizes the top-10 activating patches for representative clean and anomaly prototypes when PatchTrace trains on Patch-4. Appendix G provides additional explanation-feature statistics, prototype activations, and prototype-gallery visualizations.

G Additional Analysis of Prototype Explanations

This appendix provides additional evidence supporting the interpretability claim of PatchTrace. Our goal is not to argue that each anomaly prototype corresponds to a distinct human-nameable concept. Rather, the corrected experiments show that the anomaly bank acts as a *shared, redundant, and decision-relevant* representation of adversarial patch evidence. Specifically, we analyze: (i) the full ablation behavior on patched and clean images, (ii) the discriminative strength of explanation features derived from the prototype activations, and (iii) qualitative prototype visualizations across unseen patch types.

Table 8: Full anomaly-bank ablation results on mixed unseen patches. Reported values are mean patched probabilities.

K	**Patched Top-K**	**Patched Random-K**	**Clean Top-K**	**Clean Random-K**
0	1.0000	1.0000	0.0000	0.0000
10	1.0000	1.0000	0.0000	0.0000
20	1.0000	1.0000	0.0000	0.0000
40	0.9970	0.9980	0.0000	0.0000
60	0.8894	0.9122	0.0000	0.0000
80	0.4800	0.4882	0.0000	0.0000
100	0.4351	0.4354	0.0000	0.0000
120	0.4177	0.4244	0.0000	0.0000

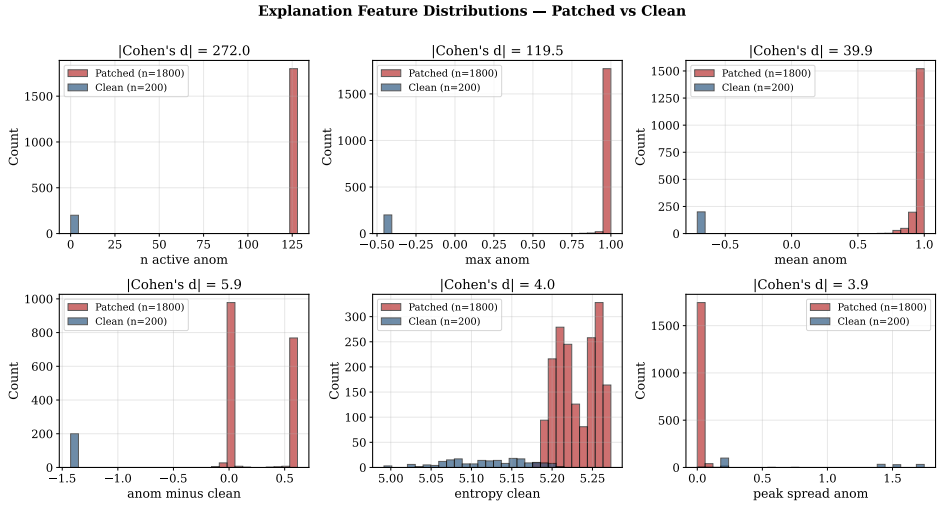


Figure 7: Feature Distribution

G.1 Full anomaly-bank ablation on patched and clean images.

We perform inference-time ablation on the anomaly prototype bank using the mixed unseen-patch benchmark described in the main paper, which contains 1,800 patched images from 9 unseen patch types and 200 clean images. For each image, we remove K anomaly prototypes and re-run the downstream detector. We consider two settings: *top-K* ablation, where the most strongly activated anomaly prototypes are removed, and *random-K* ablation, where K prototypes are sampled uniformly from the full anomaly bank.

Table 8 reports the full results. On patched images, the detector remains nearly unchanged under moderate ablation but degrades sharply once a large portion of the anomaly bank is removed. In contrast, on clean images, the patched probability remains essentially zero under both top-K and random-K ablation for all tested values of K. This confirms the main-paper claim that the anomaly bank is necessary for patched-image detection but not involved in clean-image decisions. At the same time, the close match between top-K and random-K curves at larger K values shows that the anomaly evidence is distributed across the bank, rather than concentrated in a few individually critical prototypes.

G.2 Prototype visualization across training patches.

Figure 5 visualizes the prototype patches learned when PatchTrace trains separately on each individual patch family. In each panel, the *anomaly prototypes* consistently capture localized patch-specific evidence, such as strong artificial textures, sharp boundaries, repeated high-frequency patterns, and distinctive color transitions introduced by the adversarial patch. In contrast, the *clean prototypes* focus on normal image content, including smooth background regions, natural textures, object sur-

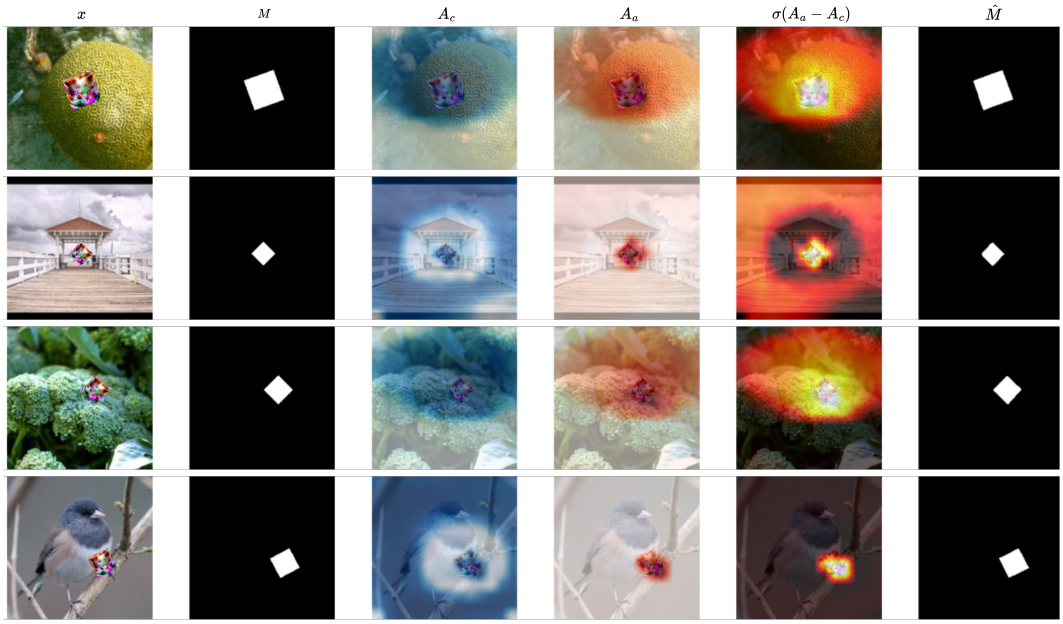


Figure 8: **Visualization of clean-bank, anomaly-bank, and PRAM activations.** For each row, x denotes the input patched image and M denotes the ground-truth patch mask. The maps A_c and A_a show the clean and anomaly prototype activations, respectively, while $\sigma(A_a - A_c)$ shows the Prototype Residual Attention Map (PRAM). The clean bank captures normal scene structure, the anomaly bank responds strongly to the adversarial region, and PRAM highlights locations where anomaly evidence exceeds clean evidence. This residual competition produces a sharper and more patch-focused localization signal than either bank alone.

Table 9: Explanation-feature separation between patched and clean images on the mixed unseen-patch benchmark.

Feature	**Patched Mean**	**Clean Mean**	$\lvert d \rvert$
# active anomaly prototypes	127.5761	0.0000	271.99
Max anomaly activation	0.9966	-0.4514	119.48
Mean anomaly activation	0.9693	-0.7072	39.94
Anomaly − clean gap	0.2629	-1.4226	5.90
Peak spread of anomaly peaks	0.0045	0.8467	3.87
Entropy of anomaly map	4.8365	5.1841	1.98
Max clean activation	0.7338	0.9712	0.83
Mean clean activation	0.4019	0.9547	0.87
# active clean prototypes	144.8011	256.0000	0.92
Entropy of clean map	5.2311	5.1202	3.99

faces, and low-frequency scene structure. This separation shows that PatchTrace does not mix normal and adversarial evidence in a single prototype space; instead, it learns two functionally distinct banks, where the anomaly bank specializes to patch-sensitive cues and the clean bank models non-adversarial visual structure.

This qualitative behavior also supports the design of PatchTrace. Because the clean and anomaly banks respond to different visual evidence, the detector can compare them explicitly at inference time rather than relying on an implicit separation through classifier weights alone. As a result, patched images activate anomaly prototypes more strongly around the attacked region, while clean images mainly activate clean prototypes and keep anomaly responses low. Across different training patches, the anomaly bank adapts to the corresponding patch family, but the clean bank remains focused on normal scene structure, which further highlights the intended role of the dual-bank design.

G.3 Explanation features are highly discriminative.

In Figure 7 and Table 9 we next analyze whether the prototype activations themselves carry sufficient information to distinguish patched from clean images. From the clean and anomaly similarity maps, we extract ten explanation features, including the number of active anomaly prototypes, maximum anomaly activation, mean anomaly activation, anomaly-minus-clean activation gap, entropy of the anomaly map, and corresponding clean-bank statistics. The resulting distributions are strongly separated between patched and clean inputs. For example, patched images activate 127.58 of the 128 anomaly prototypes on average, while clean images activate none, yielding a Cohen's d of 271.99. The maximum anomaly activation achieves Cohen's $d = 119.48$, and the mean anomaly activation achieves Cohen's $d = 39.94$. A random forest trained only on these explanation features reaches AUROC $= 1.0000$ under 5-fold cross-validation.

These results reinforce the bank-level interpretation of PatchTrace. Even though the anomaly prototypes are not individually specialized, their collective activation pattern is highly informative and fully aligned with the detector's decision boundary. Thus, the explanation is not merely visual support; it contains sufficient discriminative structure to recover the patched-versus-clean decision on its own.

G.4 Visualization of Clean Evidence, Anomaly Evidence, and PRAM

Figure 8 visualizes the internal evidence maps used by PatchTrace for localization. For each example, x denotes the input image and M denotes the ground-truth patch mask. The map A_c shows the clean-bank activation, A_a shows the anomaly-bank activation, and $\sigma(A_a - A_c)$ shows the output of the Prototype Residual Attention Map (PRAM). The clean activation map responds to normal scene structure around the patch and the surrounding background, whereas the anomaly activation map concentrates more strongly on the adversarial region. PRAM sharpens this contrast by suppressing clean evidence and retaining locations where anomaly evidence dominates, which yields a localized residual map that aligns more closely with the true patch region.

These examples illustrate why PRAM improves localization over direct anomaly activation alone. The anomaly bank often activates on both the patch and nearby context, especially when the patch appears on textured or semantically rich regions. The clean bank captures part of this normal context, and the residual $\sigma(A_a - A_c)$ removes much of that shared response. As a result, PRAM produces a more compact and patch-focused localization signal, which the decoder can convert into a more accurate binary mask.

H Computational Cost

Table 10 compares the per-image inference cost of PatchTrace and three localization-capable baselines on a single NVIDIA H200 GPU. We use the same timing protocol for every method, with 10 warmup iterations, 20 timed iterations, and CUDA synchronization before and after each call. We measure each method at its native input resolution. Under this protocol, PatchTrace processes a 224×224 image in 8.95 ± 0.05 ms with 33.4M parameters, 5.09 GFLOPs, and 0.56 GB peak GPU memory.

Among the baselines, NAPGuard achieves the lowest latency at 4.53 ± 0.07 ms with 7.0M parameters, but it does so at substantially weaker localization accuracy than PatchTrace. PAD requires $5{,}369.63 \pm 11.72$ ms per image, mainly because SAM ViT-L performs mask generation at its native 1024^2 resolution. DIFFender requires 120.68 ± 20.15 ms and 1,066.2M parameters due to its iterative diffusion-based defense pipeline. Relative to PatchTrace, PAD is approximately $600\times$ slower, and DIFFender is $13.5\times$ slower with $31.9\times$ more parameters.

These results place PatchTrace at the most favorable operating point among the compared methods. It remains only moderately slower than NAPGuard while delivering much stronger localization, and it is orders of magnitude cheaper than the SAM- and diffusion-based defenses. PatchTrace therefore provides the best practical balance between computational efficiency and cross-patch localization accuracy.

Table 10: Computational cost on a single NVIDIA H200 GPU. Each method is measured at its native input resolution (PatchTrace: 224, NAPGuard: 640 [YOLOv5], PAD: 1024 [SAM internal], DIFFender: 512 [SD internal]) using 10 warmup + 20 timed inference runs with CUDA synchronization before and after every call. Latency is reported as median ± std over the 20 timed runs. "Params (M)" is the total parameter count of every learned component on the inference path (e.g., for PAD this is the SAM ViT-L backbone, since the MI/CD heatmap step is parameter-free). FLOPs are reported only for unified-network methods; PAD and DIFFender are multi-stage pipelines and an iterative diffusion sampler respectively, neither of which is traceable by fvcore. Best value per column in bold.

Method	**Params (M)**	**FLOPs (G)**	**Latency (ms)**	**Peak Mem (GB)**
NAPGuard (Wu et al., 2024)	**7.0**	7.88	**4.53 ± 0.07**	**0.09**
PAD (Jing et al., 2024)	312.3	—	5 369.63 ± 11.72	1.20
DIFFender (Wei et al., 2025)	1 066.2	—	120.68 ± 20.15	2.43
PatchTrace (ours)	33.4	**5.09**	8.95 ± 0.05	0.56