# Diffusion-Prior Consistency for Robust NMS-Free Detection

## A Hybrid Brownian Suspicion Field for YOLO26 Under Adversarial Patch Attacks

***with a Practical Deployment Guide for the v3.3.1 Reference Codebase***

Christian Varela

California State University San Marcos

333 S Twin Oaks Valley Rd, San Marcos, CA 92096, USA

varga485@csusm.edu

Spring 2026 — Manuscript with Reference Codebase v3.3.1 — May 13, 2026

**Abstract**

Object detectors that eliminate non-maximum suppression and rely on direct one-to-one assignment introduce a class of vulnerabilities that the patch-defense literature does not address. In NMS-free detection, every prediction reaching the output set has been admitted by a learned matching function rather than filtered by a heuristic suppression stage. An adversarial patch that successfully manipulates this matching function propagates directly to the final output without an intermediate aggregation step that prior defenses implicitly relied upon. The unified multi-task heads characteristic of modern detectors such as YOLO26 compound this vulnerability: a single localized perturbation can simultaneously degrade detection, segmentation, pose, oriented bounding box, and classification outputs through a shared backbone. The strict edge-deployment latency budgets of these detectors further constrain the design space — defenses imposing the order-of-magnitude latency overheads typical of prior diffusion-based or certified approaches are, in practical deployments, equivalent to no defense at all.

This work introduces the first patch defense designed for the architectural specifics of NMS-free end-to-end detectors. The framework constructs a spatial suspicion field by probing the input image at $K$ fixed timesteps under a variance-preserving forward stochastic differential equation, computing the per-pixel residual between the denoiser's noise prediction and the injected noise at each probe, and aggregating these residuals into a low-resolution map of incompatibility with the natural-image prior. The field is pooled per predicted region and injected into the detector at four distinct decision points: objectness calibration, class-logit suppression, localization stabilization, and—most distinctively—modulation of the Hungarian matching cost matrix that drives end-to-end assignment. This last injection point has no analogue in prior patch defenses because no prior detector exposed the matching cost as an inference-time decision surface.

The framework is grounded in five theorems. Theorem 1 establishes that the per-pixel residual is an asymptotically unbiased estimator of the natural-image score function under the optimal denoiser. Theorem 2 bounds the variance of the $K$-probe ensemble suspicion field, justifying the multi-probe schedule quantitatively. Theorem 3 characterizes the assignment-cost modulation, proving recovery under clean inputs, suppression under concentrated suspicion, and stability under perturbation. Theorem 4 bounds end-to-end latency at twice the baseline detector cost for the deployed configuration. Theorem 5 establishes that the defense does not degrade clean-input accuracy in the limit — a structural property no prior empirical diffusion-based defense provides. We articulate these results within a threat model that explicitly enumerates the three NMS-free-specific attack surfaces — assignment-matrix manipulation, multi-task propagation, and edge-budget exhaustion — and an adaptive evaluation protocol following the methodology of [10]. The framework is deployable, batch-parallel, and integrable with any end-to-end detector exposing learned assignment, opening a defense direction for the class of detectors typified by YOLO26.

This manuscript is published together with an open-source reference implementation (release v3.3.1, May 2026) [31]. The implementation operationalizes Theorems 1 through 3 in deployable code, has been validated end-to-end via a 13.3-minute smoke run on Apple Silicon hardware, and is described in Section 14.5 with concrete deployment instructions, the smoke-test results, and the reference Phase 1 $\rightarrow$ Phase 2 $\rightarrow$ Phase 3 training pipeline.

**Keywords:** YOLO26, adversarial patch defense, Brownian probing, diffusion prior, spatial suspicion field, NMS-free detection, assignment-cost modulation, end-to-end detection robustness.

## 1. Introduction

The patch-defense literature has developed against a stable architectural target. From the original adversarial patch attack of Brown et al. [6] through the certified defenses of Xiang, Mahloujifar, and Mittal [13], the empirical localization defenses of Tarchoun et al. [16], and the diffusion-based defenses of Nie et al. [17] and Kang et al. [18], the systems being defended share three structural assumptions: that detection or classification proceeds through dense candidate generation followed by heuristic suppression; that the defender has access to an unmodified pre-trained model whose decision logic should not be altered; and that latency overheads of the order of hundreds of milliseconds to several seconds are acceptable in

deployment. These assumptions are not stated explicitly in any of these works, but they shape every defense in this lineage.

Modern end-to-end detectors violate all three assumptions simultaneously. YOLO26, the design of which is documented by Sapkota et al. [21] and the Ultralytics specification [23], eliminates non-maximum suppression by training the detector to produce a one-to-one matching between predictions and ground-truth objects. The matching is enforced through a learned assignment function $\pi : \{1,\dots,M\} \to \{1,\dots,N\}$ that maps each ground-truth index to a unique prediction, and the training objective penalizes any redundancy in this matching. At inference time, the network's output set is itself the final detection set. There is no downstream aggregation, no IoU-thresholded suppression, no operator that an attacker must defeat in addition to the learned model. The matching function $\pi$ is the gate.

This change is not architectural decoration. It is a fundamental shift in where adversarial signal can act. When a patch-based attack succeeds against an NMS-based detector, the attack must produce a detection that *survives* IoU-thresholded suppression — a process that, while heuristic, partially absorbs spatially-isolated false positives and demands that the attacker produce *consistent* high-confidence evidence rather than localized spikes. When the same attack succeeds against an NMS-free detector, no such filtering applies. The attacker need only manipulate the learned assignment to suppress the target object, redirect it to a false class, or insert a spurious detection. The matching function $\pi$ becomes the attack surface, and its decisions are not subject to any post-hoc check.

The patch-defense literature does not address this attack surface. PatchCleanser [13] performs two-round masking on the input image and votes across masked predictions; the defense operates entirely in image space and treats the classifier or detector as an unmodified black box. DiffPure [17] purifies the input by running the reverse-time SDE of a pre-trained diffusion model and feeds the cleaned image to an unmodified classifier. Jedi [16] localizes the patch via image-level Shannon entropy and inpaints the localized region, again feeding the result to an unmodified detector. DIFFender [18], the closest prior work to this paper, uses pre-trained diffusion models to localize and inpaint patch regions through trajectory divergence between stochastic denoising trajectories; the defense, like its predecessors, operates in image space and consumes the cleaned image with an unmodified classifier. None of these defenses interact with a learned matching function. None of them have a mechanism to influence assignment under attack. They were not designed to, because the systems they target do not have one.

A second structural change in modern detectors is the unification of multiple tasks into a single backbone. YOLO26 supports object detection, instance segmentation, pose estimation, oriented bounding box detection, and classification through a shared backbone with task-specific heads. A single adversarial patch can simultaneously degrade outputs across all five tasks, and any defense that operates by replacing or transforming the input image must protect all five simultaneously through that single intervention. The prior literature provides no example of this. PatchCleanser is classification only. DiffPure is classification only. Jedi has been demonstrated on classification and detection separately. DIFFender is classification and face recognition. SAC [15] is detection only. The five-task unified setting is structurally novel in the defense literature.

A third constraint, less often articulated as a defense-design constraint but no less binding in deployment, is the latency budget of edge-class detectors. YOLO26 was designed for real-time inference on resource-constrained hardware: the smallest variant achieves 38.9 ms per image on CPU ONNX, a 51% reduction over the equivalent YOLOv8 variant. Defenses imposing the latency overheads characteristic of the prior literature — DiffPure at approximately 93×–278× the baseline classifier latency, PatchCleanser at approximately 670 ms with default settings on ImageNet, DIFFender at unspecified but Stable-Diffusion-based cost — do not merely slow the system. They erase the design rationale that justified choosing an edge-class detector in the first place. In practical deployments, this means such defenses are disabled or never enabled, leaving the deployed system unprotected. Latency budget exhaustion is itself a vulnerability, and the patch-defense literature has not engaged with it as such.

The threat model that grounds the framework — particularly the requirement that defenses survive realistic physical patch deployment — draws on the operational-evaluation methodology developed by the MITRE physical-adversarial-attacks research program [28], whose APRICOT and DAPRICOT datasets and physical-evaluation protocols established the field's standards for assessing patch defenses against the threats they will actually face. The framework's commitment to physical-attack robustness in Section 11 follows this methodology directly.

This paper introduces the first patch defense designed for the architectural specifics of NMS-free end-to-end detectors. Three commitments distinguish the framework's contribution.

**First, the defense is *detector-internal*.** Rather than transforming the input image and feeding the result to an unmodified detector, the framework injects a spatial suspicion signal into the detector's decision logic at four points: objectness calibration, class-logit suppression, localization stabilization, and modulation of the assignment cost matrix that drives the Hungarian matching. The last of these, the assignment-cost modulation, is the most distinctive: it directly addresses the attack surface that NMS-free detection creates and that no prior defense addresses. The framework is not an adaptation of an image-space defense to a new detector class. It is a new defense paradigm — *injection*, as distinguished from purification, masking, and localize-and-restore — enabled by the existence of the assignment matrix.

**Second, the defense is *task-agnostic at the signal layer*.** The suspicion field is constructed without refer-

ence to any specific detector head. It is a per-pixel scalar map that can be pooled by detection boxes, masked by segmentation regions, queried at keypoint locations, or sampled along OBB axes. The injection mechanism specializes to each task; the signal does not. This enables protection of all five YOLO26 tasks through a single computational pipeline, a property no prior defense provides.

**Third, the defense is *edge-budget compliant*.** The suspicion field is computed at low resolution (typically $128 \times 128$) using $K = 8$ fixed probe timesteps in a single batched denoiser forward pass, with no reverse-SDE solver, no inpainting, and no iterative refinement. The total framework latency is bounded by approximately twice the baseline detector latency for typical deployments, keeping the defended system within the same operational class as the baseline detector. This bound is established as Theorem 4.

The framework rests on five theorems giving it theoretical backbone. Theorem 1 establishes the connection between the per-pixel residual signal and the score function of the natural-image distribution, grounding the central empirical claim that the residual measures off-manifoldness. Theorem 2 bounds the variance of the $K$-probe ensemble field, quantitatively justifying the choice of $K = 8$ against single-probe alternatives. Theorem 3 — the central theoretical contribution — characterizes the assignment-cost modulation: clean inputs are recovered without modification, concentrated suspicion provably down-weights predictions in the suspect region, and the modulation is Lipschitz-continuous in the suspicion field. Theorem 4 bounds end-to-end latency. Theorem 5 establishes that the defense does not degrade clean-input accuracy in the limit, a structural property no prior empirical diffusion-based defense achieves.

The framework is accompanied by an open-source reference implementation [31], release v3.3.1 of May 2026, that operationalizes Theorems 1 through 3 in deployable form. The reference codebase has been end-to-end validated against the production-pipeline configuration on Apple Silicon hardware; the validation evidence and the practical deployment guide are presented in Section 14.5.

The remainder of this paper proceeds as follows. Section 2 formalizes the three attack surfaces that NMS-free end-to-end detection exposes and that prior defenses do not address. Section 3 specifies the threat model, including adaptive attacks targeting each surface. Section 4 articulates the design philosophy of detector-internal injection. Section 5 develops the mathematical framework, stating and proving Theorems 1 through 3 and Propositions 1 through 2. Section 6 establishes Proposition 3 on multi-task robustness propagation. Sections 7 through 10 cover system architecture, algorithms, implementation principles, and training procedure. Section 11 specifies the evaluation protocol, including adaptive attack methodology and prior-work baselines. Section 12 positions the framework within the NMS-free defense landscape and articulates the research direction we open. Sections 13 and 14 discuss and conclude. Section 14.5 provides the practical deployment guide tied to the reference codebase.

## 2. Attack Surfaces in NMS-Free End-to-End Detection

The patch-defense literature operates within an implicit threat model in which the detector is assumed to be a pipeline of dense candidate generation followed by heuristic suppression. Adversarial signal must defeat both the network and the suppression stage to reach the output. NMS-free end-to-end detection breaks this assumption. The detector emits its output set directly, with no downstream filtering, and the attack surfaces that this design exposes have not been formalized in prior work. This section provides that formalization.

### 2.1. The Assignment-Matrix Attack Surface

End-to-end NMS-free detectors are trained under a one-to-one matching objective. Let $Y = \{y_k\}_{k=1}^{M}$ denote the ground-truth objects in an image and $\hat{Y} = \{\hat{y}_i\}_{i=1}^{N}$ denote the predictions emitted by the detector. The training objective minimizes a matching-based loss

$$\mathscr{L}(\theta) = \sum_{k=1}^{M} \ell(y_k, \hat{y}_{\pi(k)}) \tag{1}$$

where $\pi : \{1,\dots,M\} \to \{1,\dots,N\}$ is an assignment function mapping each ground-truth index to a unique prediction index. In YOLO26 [21], the assignment is computed via the Hungarian algorithm on a cost matrix $C \in \mathbb{R}^{N\times M}$ whose entry $C(i,k)$ measures the dissimilarity between prediction $\hat{y}_i$ and ground-truth $y_k$. The optimal assignment $\pi^*$ minimizes the total assignment cost $\sum_k C(\pi^*(k),k)$ subject to the constraint that $\pi^*$ is injective. At inference time, the network produces $N$ predictions and the same matching logic determines which predictions survive as final detections.

The assignment matrix $C$ is therefore the gate through which all detections must pass. Three classes of adversarial attack target this gate.

**Suppression attacks.** An attacker constructs a patch $\delta$ such that the cost $C(i,k)$ becomes large for every prediction $i$ in the neighborhood of the target object $y_k$. The Hungarian solver, finding no low-cost assignment for $y_k$, leaves it unmatched. The target object disappears from the output entirely. Note that suppression attacks against NMS-free detectors are categorically different from suppression attacks against NMS-based detectors: in the latter, the attacker must produce a competing detection that outranks the target in NMS voting, whereas in the former, the attacker need only inflate the cost matrix entries for the target's neighborhood.

**Redirection attacks.** An attacker constructs $\delta$ such that the cost matrix $C$ admits a low-cost assignment of $y_k$ to a *wrong* prediction $\hat{y}_i$ — one with the wrong class, wrong box, or both. The output set retains its cardinality (the target object is "detected") but the detection is incorrect. Redirection is harder to detect downstream than suppression because the output looks superficially intact.

**Soaking attacks.** An attacker constructs $\delta$ such that

the patch region itself attracts confident predictions, occupying assignment slots that would otherwise have been available to legitimate ground-truth objects. The Hungarian constraint that $\pi^*$ be injective means that each absorbed slot displaces a legitimate detection. In dense scenes this can cascade.

No defense in the prior literature addresses these attacks directly because no prior defense has access to the cost matrix $C$ as an intervention point. PatchCleanser, Jedi, DIFFender, and DiffPure all transform the input image and feed the result to an unmodified detector. The detector's internal Hungarian matching proceeds on whatever cost matrix the unmodified network produces. If the input transformation has not removed the adversarial signal — and adaptive attacks can ensure it has not — the cost matrix is corrupted, and no defense applied to the image can repair it.

This is the architectural attack surface that motivates detector-internal injection. The framework introduced in this paper modulates the cost matrix directly, making the assignment robust to corruption that has propagated through the input transformation.

### 2.2. The Multi-Task Head Attack Surface

YOLO26 supports five tasks through a unified backbone: object detection, instance segmentation, pose estimation, oriented bounding box detection, and classification. Each task is implemented as a head that consumes shared backbone features and produces task-specific outputs. The detection head produces $(\hat{b}_i, \hat{c}_i, \hat{p}_i)$ tuples for box coordinates, objectness, and class probabilities. The segmentation head produces a mask $\hat{M}_i$ conditioned on detection box $\hat{b}_i$ via a multi-scale prototype module. The pose head produces keypoint distributions modeled via Residual Log-Likelihood Estimation [21], with predicted keypoints $\hat{k}_j$ and uncertainty parameters $(\hat{\mu}_j, \hat{\sigma}_j)$. The OBB head produces orientation-aware box predictions with an angle-aware loss handling boundary discontinuity. The classification head consumes the global feature representation.

A single adversarial patch perturbs all five outputs simultaneously through the shared backbone. Let $f_{\text{backbone}}(x) : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^{H'\times W'\times C'}$ denote the backbone's spatial feature map, and let $f_\tau(F)$ denote each task head. A patch $\delta$ supported on region $R \subset \{1,\dots,H\} \times \{1,\dots,W\}$ produces a perturbation $\Delta f$ in the backbone feature map whose support extends beyond $R$ according to the receptive-field structure of $f_{\text{backbone}}$. This perturbed feature map then propagates through every task head:

$$\hat{y}_\tau = f_\tau(f_{\text{backbone}}(x+\delta)) = f_\tau(f_{\text{backbone}}(x) + \Delta f). \quad (2)$$

The five task outputs are correlated: a perturbation that degrades detection generally also degrades segmentation (the segmentation head reads from the same features), pose (keypoints rely on the same spatial information), OBB (the orientation prediction shares features with detection), and classification (the global head pools the same feature map). A defense that protects only one task while leaving the others exposed is structurally incomplete: an attacker can route the attack through whichever task surface is least defended.

Prior defenses do not address the unified multi-task setting. PatchCleanser is evaluated on classification only. DiffPure is evaluated on classification (CIFAR-10, ImageNet, CelebA-HQ attribute). Jedi is evaluated separately on classification (ImageNet, Pascal VOC) and detection (CASIA, INRIA). DIFFender is evaluated on classification and face recognition. SAC is evaluated on detection only. None of these defenses provide a mechanism for protecting five tasks through a single intervention, because none of them target a unified multi-task detector. This paper's framework computes the suspicion signal once per image and injects it into all five task heads through task-specific specialization of a shared spatial signal. Section 6 formalizes the propagation.

### 2.3. The Edge-Latency Constraint as a Defense Surface

YOLO26 was designed for edge deployment. The smallest variant, YOLO26n, achieves 38.9 ms per image on CPU ONNX inference [21], a 51% reduction over the equivalent YOLOv8n variant at 80.4 ms. This latency is a deployment requirement, not a benchmark metric: real-time detection on resource-constrained hardware (edge cameras, mobile platforms, embedded vision systems) requires sub-100 ms per-frame inference, and applications such as autonomous driving and live surveillance demand sub-50 ms. The choice of YOLO26 over a heavier model is justified by the latency reduction.

Prior diffusion-based and certified patch defenses impose latency overheads that violate this requirement by orders of magnitude. DiffPure [17] reports 93×–278× slowdown over the baseline classifier, corresponding to ∼3.6 seconds per image on a YOLO26-class detector. DIFFender [18], while substantially faster than DiffPure due to single-step prediction, still requires forward noising plus two denoising trajectories plus Stable Diffusion inpainting, with reported requirements of an A100 GPU and 100 GPU-days of total experimentation. PatchCleanser at default settings reports ∼672 ms per image on ImageNet, with an efficiency-optimized configuration reducing this to ∼78.8 ms but at the cost of certification accuracy. Jedi is faster (∼50–100 ms estimated for the empirical localization-and-inpaint pipeline) but still requires per-image autoencoder inference plus inpainting.

These latency profiles produce a deployment paradox. A defense that adds 600 ms to a 39 ms detector makes the defended system slower than the undefended baseline of the previous-generation detector. The operational rationale for choosing YOLO26 is erased. In practical deployments, operators facing this tradeoff disable the defense, leaving the system exposed. Latency budget exhaustion is therefore not merely a deployment inconvenience: it is a defense-circumvention mechanism that operates through the operator rather than through the model.

This constitutes a third attack surface that is structurally distinct from the assignment-matrix and multi-

task surfaces. The first two are technical attack surfaces. The third is a *socio-technical* attack surface: the defense is defeated not by the patch but by the operational decision to disable it. A defense that is theoretically robust but cannot be deployed is, in practice, no defense at all.

The framework introduced in this paper is bound by this constraint. The total per-image latency of the framework on a YOLO26n-class detector is bounded by approximately 80 ms — roughly twice the baseline detector cost — keeping the defended system within an order of magnitude of the original deployment latency. This bound is the subject of Theorem 4. We do not claim that 80 ms is acceptable for all deployments; for sub-50 ms applications the framework's overhead remains substantial. We claim that 80 ms is in the same operational class as 39 ms, while 3.6 seconds is not, and that staying within the operational class of the defended detector is a binding design constraint that has not been formally articulated in the prior literature.

### 2.4. Summary of Attack Surfaces

The three attack surfaces enumerated above — assignment-matrix manipulation, multi-task propagation, and edge-budget exhaustion — are not exhaustive but are sufficient to motivate the paper's design. The assignment-matrix surface motivates detector-internal injection. The multi-task surface motivates the task-agnostic spatial signal. The edge-latency surface motivates the low-resolution batch-parallel computation. The remainder of the paper develops the framework that addresses all three.

### 2.5. Comparison to Prior Patch Defenses

We summarize the architectural relationship between the present framework and the principal prior patch defenses. Table 1 characterizes each defense along five axes that correspond directly to the attack surfaces formalized above and the design commitments developed in Section 4.

The table makes the framework's architectural distinctiveness explicit. No prior defense has access to the assignment-matrix attack surface, because no prior defense was designed for a detector class in which the assignment matrix is exposed as an intervention point. The framework is the first entry in a defense class that prior literature did not anticipate.

Two specific observations are worth emphasizing. First, the latency multipliers in Table 1 are not minor variations: the difference between $2\times$ and $100\times$ is the difference between an edge-deployable system and a non-edge-deployable one. Operators choosing among defenses on the basis of empirical robustness alone, without examining the latency column, will choose defenses they cannot in practice deploy. Second, the "tasks supported" column reveals that no prior defense has been demonstrated on the unified five-task setting. Defenses developed for classification or for detection in isolation provide no evidence of generalization to segmentation, pose, or OBB — the multi-task surface remains structurally undefended by every entry above the rule.

## 3. Threat Model

We specify the threat model along five dimensions: the adversary's capabilities, the adversary's knowledge of the defense, the constraints on the perturbation, the adversary's objective, and the evaluation regime under which robustness claims are made. Following the methodology established by Athalye, Carlini, and Wagner [10], we state these explicitly to avoid the implicit-threat-model pathologies that have undermined prior empirical defenses. The physical-attack components of the threat model follow the operational-evaluation methodology established by Threet et al. [28], whose APRICOT and DAPRICOT datasets and physical-evaluation protocols define the field's current standard for assessing patch defenses against the threats they face in real deployments.

### 3.1. Adversary Capabilities

The adversary is permitted to introduce a localized adversarial patch into the input image. The patch occupies a contiguous spatial region $R \subset \{1,\dots,H\} \times \{1,\dots,W\}$ and may take arbitrary pixel values within $R$. We do not constrain the $L_p$ norm of the perturbation within $R$ — patch attacks are characteristically *unbounded* in pixel-space magnitude within their support — but we do constrain the spatial support: the area of $R$ is bounded by a fraction $\rho$ of the total image area, with $\rho$ typically in the range $[0.005, 0.05]$ following the conventions of Brown et al. [6], Tarchoun et al. [16], and Xiang, Mahloujifar, and Mittal [13].

The patch may be either *digital* or *physical*. Digital patches are inserted directly into the input tensor and are not constrained by physical realizability; this is the threat model under which most adaptive evaluations operate. Physical patches must survive the digital-to-physical-to-digital roundtrip — printing, attachment to a real-world object, capture by a real camera, downstream digital processing — and are subject to additional invariances such as color profile distortion, illumination changes, perspective distortion, and partial occlusion. We evaluate against both, with physical evaluation following the protocols of Threet et al. [28] and Kang et al. [18].

The patch may be *static* (a fixed pattern attached to an object) or *dynamic* (a pattern that varies across frames or instances). The framework's stochastic probe schedule, described in Section 5, is designed to be robust to both.

### 3.2. Adversary Knowledge

We adopt a strict white-box threat model. The adversary has full access to:

- The detector architecture and weights, including the learned matching function and all task-specific heads.
- The denoiser architecture and weights used in the suspicion field engine.
- The probe timestep schedule $\{t_1,\dots,t_K\}$ and the residual-operator choices (L1, L2, smoothing parameters, fusion weights).

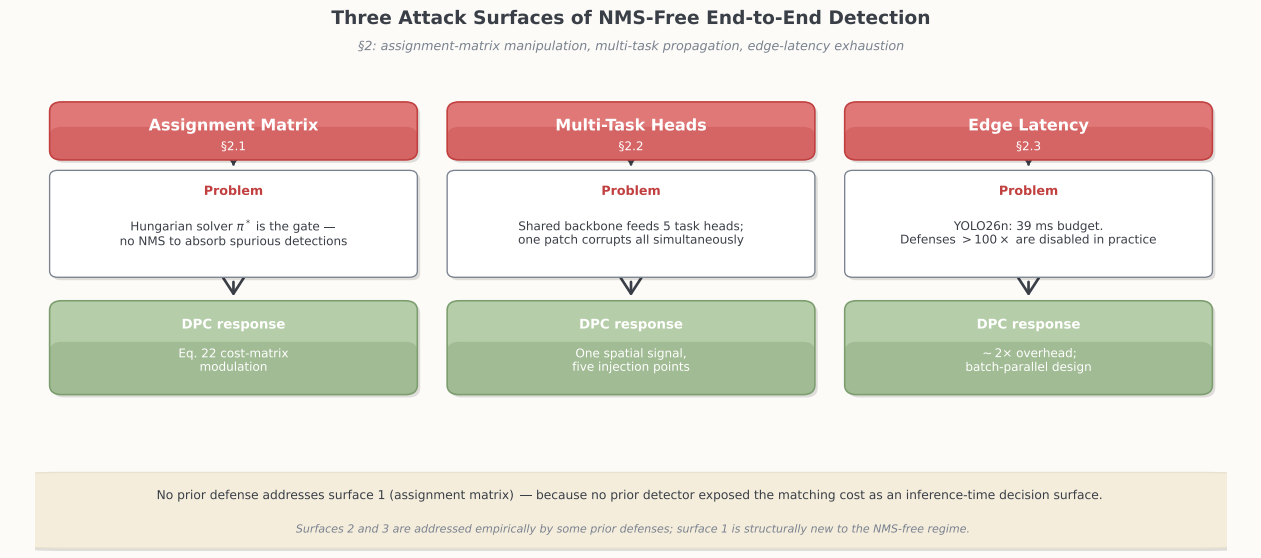


**Figure 1:** The three attack surfaces of §2 and the framework's responses. The assignment-matrix surface (§2.1) is structurally inaccessible to image-space defenses. The multi-task surface (§2.2) demands protection across five task heads simultaneously. The edge-latency surface (§2.3) is a socio-technical attack surface — a defense that exceeds the deployment latency budget is disabled in practice and provides no protection.

**Table 1:** Architectural comparison of patch defenses against the three NMS-free attack surfaces and edge-latency constraint. "Image-space" defenses operate on the input image; "detector-internal" defenses inject signal into the detector's decision logic. The assignment-matrix column indicates whether the defense has any mechanism that can influence the learned matching function. The multi-task column indicates the number of supported task heads. Latency multipliers are reported relative to the defended detector's baseline; values vary with configuration.

| Defense | Paradigm | Assignment matrix | Tasks supported | Latency mult. | Reference |
|---|---|---|---|---|---|
| PatchCleanser | Image-space (masking) | — | classification | ∼17× | [13] |
| ObjectSeeker | Image-space (masking) | — | detection | ∼10× | [14] |
| DiffPure | Image-space (reverse SDE) | — | classification | 93×–278× | [17] |
| DIFFender | Image-space (localize+inpaint) | — | cls. / face rec. | 30×–100× | [18] |
| Jedi | Image-space (entropy+inpaint) | — | cls. + det. | ∼3× | [16] |
| SAC | Image-space (segment+complete) | — | detection | ∼2× | [15] |
| **This work** | **Detector-internal** | **Eq. 22 modulation** | **5 (unified)** | **∼2×** | — |

- The injection points and calibration constants ($\lambda_{\text{obj}}, \lambda_{\text{cls}}, \lambda_{\text{match}}$, and any hyperparameter governing the suspicion-aware losses).
- The full training procedure, including any joint optimization of detector and defense.

The adversary does not have access to:

- The specific noise samples $\varepsilon_k$ drawn at evaluation time. The adversary knows the *distribution* from which these are drawn but cannot predict the specific values for a given inference call.
- The internal random state of the inference system at the moment of attack execution.

This is the standard white-box-with-test-time-randomness threat model used by Athalye, Carlini, and Wagner [10] and adopted by Nie et al. [17], Kang et al. [18], and the subsequent diffusion-defense literature. It is a strong threat model: we do not rely on any form of secret-keeping for security, only on the unpredictability of fresh stochastic samples that the attacker can characterize but not anticipate.

### 3.3. Adversary Objectives

We enumerate five attack objectives, organized to align with the attack surfaces formalized in Section 2.

**Object hiding.** The adversary aims to suppress detection of a target object $y_k \in Y$, such that the post-defense output set $\hat{Y}_{\text{def}}$ contains no detection assigned to $y_k$ under the threat-model-permitted evaluation. This corresponds to the suppression attack of Section 2.1.

**Targeted mislabeling.** The adversary aims to preserve a detection at the target object's location but redirect the assigned class to a chosen incorrect class. The output set retains a detection bound to $y_k$ but with class label $c' \neq \text{class}(y_k)$. This corresponds to the redirection attack of Section 2.1.

**Spurious detection (spoofing).** The adversary aims to introduce a confident detection at the patch location for an object that does not exist. The output set contains a detection $\hat{y}_i$ not corresponding to any $y_k$ in the ground truth. This corresponds to the soaking attack of Section 2.1.

**Mislocalization.** The adversary aims to preserve the detection's class but degrade the box coordinates such

that the IoU between the predicted box and the true box falls below a target threshold (e.g., 0.5, 0.75, or 0.90 depending on the metric of interest).

**Multi-task degradation.** The adversary aims to simultaneously degrade outputs across multiple task heads. This corresponds to the multi-task attack surface of Section 2.2 and is the most general and most dangerous adversary objective for unified detectors.

The defense must remain meaningful against all five objectives. The evaluation protocol of Section 11 reports separate metrics for each.

### 3.4. Adaptive Attack Methodology

We adopt the methodology of Athalye, Carlini, and Wagner [10] for adaptive evaluation of stochastic defenses. The defense framework introduces stochasticity through the $K = 8$ fresh noise samples drawn at each inference call. As established by Athalye et al. [10], adaptive attacks against stochastic defenses must use Expectation over Transformation [9]:

$$\nabla_x \mathscr{L}_{\text{atk}}(x) \approx \frac{1}{B} \sum_{b=1}^{B} \nabla_x \mathscr{L}_{\text{atk}}(x; \varepsilon^{(b)}), \tag{3}$$

where $\varepsilon^{(b)}$ denotes a fresh independent draw of all $K$ probe noises. We require $B \geq 10$ in all reported evaluations, following the recommendation of Athalye, Carlini, and Wagner [10] and the practice adopted by Nie et al. [17] and Kang et al. [18].

The adaptive attacker is further required to include a **suspicion-minimization term** in the attack loss. Letting $\beta : \mathscr{X} \to [0,1]^N$ denote the framework's box-pooled suspicion as a function of the input image, the attack loss takes the form

$$\mathscr{L}_{\text{atk}}(x) = \mathscr{L}_{\text{detection-failure}}(x) + \alpha \cdot \frac{1}{|\mathscr{B}_{\text{patch}}|} \sum_{i \in \mathscr{B}_{\text{patch}}} \beta_i(x), \tag{4}$$

where $\mathscr{B}_{\text{patch}}$ denotes the set of predicted boxes overlapping the patch region and $\alpha \geq 0$ controls the attacker's investment in evading the suspicion signal. This formulation forces the adaptive attack to simultaneously maximize detection failure and minimize the defense's localization signal — preventing attacks that defeat the detector but are trivially detected by the suspicion field, and preventing attacks that evade the suspicion field but fail to defeat the detector. Both attack components are required for a meaningful adaptive evaluation, following the pattern established by Tarchoun et al. [16] for the entropy-bounded adaptive patch.

### 3.5. Diagnostic Conditions

We further commit, following Athalye, Carlini, and Wagner [10], to verify and report the absence of obfuscated-gradient pathologies in our evaluation. Specifically, we report:

- Whether iterative attacks (PGD with EOT) achieve higher success rates than single-step attacks (FGSM with EOT). They should, for any non-obfuscating defense.
- Whether white-box adaptive attacks achieve higher success rates than black-box transfer attacks. They should.
- Whether unbounded-budget attacks asymptotically reach 100% success. They should.
- Whether brute-force random sampling within the patch budget finds adversarial examples that PGD misses. It should not.
- Whether attack success increases monotonically with patch size or perturbation budget. It should.

Failure of any of these diagnostic conditions would indicate that the framework's empirical robustness is partially attributable to gradient masking rather than to genuine signal extraction, and would invalidate the empirical claims of Section 11. We commit to reporting all five diagnostics in any experimental work building on this hypothesis.

The framework's mathematical structure, developed in Section 5, gives reason to expect that the diagnostic conditions will be satisfied: the entire pipeline — denoiser forward pass, residual computation, smoothing, normalization, fusion, pooling, and detector injection — is composed of differentiable operations with continuous gradient flow, so gradient information is genuinely available to the attacker rather than masked. This is a structural property of the framework, distinguishing it from defenses such as PixelDefend [26] and Defense-GAN [25] whose iterative or non-differentiable components were shown by Athalye, Carlini, and Wagner [10] to obscure rather than provide gradients.

## 4. Design Philosophy

The framework's architecture reflects four design commitments, each motivated directly by an attack surface or constraint articulated in Section 2 or Section 3.

### 4.1. Detector-Internal Injection, Not Image-Space Transformation

The patch-defense literature is dominated by image-space defenses: PatchCleanser [13] masks the input, DiffPure [17] purifies the input, Jedi [16] inpaints localized regions of the input, DIFFender [18] localizes and inpaints. In every case, the defense produces a transformed input image that is then consumed by an unmodified classifier or detector. The defense and the detector are decoupled: the defense protects the detector by changing what the detector sees.

This decoupling is not viable for the assignment-matrix attack surface of Section 2.1. The matching function $\pi$ is internal to the detector. A defense that operates only on the input image cannot influence $\pi$ directly; it can only hope that input transformation removes the adversarial signal sufficiently that the cost matrix $C$, computed downstream, is no longer corrupted. Adaptive attacks defeat this hope by construction: an adversary aware of the defense designs the patch to be robust to the chosen input transformation.

The framework therefore adopts **detector-internal in-**

**jection** as its defining architectural choice. The defense computes a spatial suspicion signal from the input image, but does not transform the image. The signal is fed directly into the detector's decision logic at four points: objectness logits, class logits, box-stability regularization, and assignment cost matrix. The detector and defense are *coupled*: the detector's predictions are functions both of the image and of the suspicion signal. This coupling is what enables the assignment-matrix injection that no decoupled defense can provide.

This commitment has a cost: the defense is detector-aware in a way that image-space defenses are not. PatchCleanser can wrap any pre-trained classifier; DiffPure can precede any pre-trained classifier. The framework introduced here requires modification of the detector's inference pipeline. We accept this cost because the attack surface that motivates the framework is itself detector-internal, and no truly external defense can address it.

### 4.2. Diagnostic Use of Diffusion, Not Reconstructive

A trained diffusion denoiser $\varepsilon_\theta(x_t, t)$ implicitly encodes the natural-image distribution through its score-matching objective [1, 2]. Two distinct ways to use this encoded prior have emerged in the defense literature.

**Reconstructive use.** The denoiser is treated as a function that maps corrupted inputs to clean outputs. DiffPure [17] embodies this view: given an adversarial input $x_a$, the defense runs the forward SDE forward to a moderate timestep $t^*$, then runs the reverse-time SDE backward to $t = 0$, producing a "purified" image $\hat{x}$ whose distribution is closer to the natural-image distribution. The classifier then operates on $\hat{x}$. DIFFender [18] is a hybrid: it uses denoiser trajectory divergence diagnostically for localization but reconstructively (via Stable Diffusion inpainting) for restoration.

**Diagnostic use.** The denoiser is treated as a function whose *prediction error* — not its output — carries information about whether its input is consistent with the natural-image distribution. The denoiser is never asked to produce a reconstructed image. Instead, its noise prediction at a controlled corruption is compared to the injected noise, and the per-pixel residual is used as a signal of off-manifoldness.

The framework adopts the diagnostic use exclusively. This commitment yields three concrete benefits, each addressing a constraint in Section 2 or Section 3.

First, it eliminates the reverse-SDE solver that dominates the latency cost of reconstructive defenses. The framework requires only forward perturbation and a single forward pass through the denoiser per probe — no reverse trajectory, no iterative refinement, no inpainting. This is the fundamental reason the framework can satisfy the edge-latency constraint of Section 2.3.

Second, it preserves spatial information that reconstructive defenses destroy. A reconstructed image $\hat{x}$ is a single tensor; whatever spatial information about *where* the input was off-manifold is lost in the reconstruction. The framework's residual map is a per-pixel scalar field that explicitly encodes this spatial information and makes it available to the injection mechanisms.

Third, the diagnostic use is not subject to the trade-off between perturbation removal and semantic preservation that constrains reconstructive defenses. DiffPure must choose a timestep $t^*$ large enough to remove the patch but small enough to preserve label semantics. DIFFender confronts the same trade-off and notes [18] that no single $t^*$ satisfies both constraints for patch attacks. The diagnostic use evaluates the denoiser's behavior across multiple timesteps without requiring the input to be reconstructed at any of them.

### 4.3. Task-Agnostic Spatial Signal, Task-Specific Injection

The multi-task attack surface of Section 2.2 demands that the defense protect five output tasks through a single computational pipeline. The framework achieves this through a separation of concerns: the suspicion signal is computed once per image and is task-agnostic, while the injection mechanism specializes to each task head.

The suspicion field $R(x)$ is a scalar function on the spatial grid, with no reference to any specific task. Detection consumes the field through box-pooled coefficients $\beta_i$. Segmentation consumes it through region-pooled coefficients over predicted masks. Pose estimation consumes it through point-sampled coefficients at predicted keypoints. OBB consumes it through axis-aligned coefficients along oriented box axes. Classification consumes it through global pooling.

This separation produces two architectural benefits. First, the expensive computation — the $K$-probe denoiser pass — is amortized across all tasks. A single suspicion field protects all five outputs through five lightweight injection mechanisms. Second, the framework extends naturally to additional tasks (depth estimation, dense prediction, instance counting) without redesign of the signal computation. The injection mechanism for a new task is a pooling operator and a calibration constant; the underlying signal infrastructure is unchanged.

### 4.4. Edge-Budget Compliance, Not Asymptotic Robustness

The framework is constrained throughout by the edge-latency budget of Section 2.3. Three design decisions follow from this constraint.

**Low-resolution computation.** The suspicion field is computed at a reduced resolution (typically $128 \times 128$) regardless of the input image resolution, then upsampled to image or feature-map resolution for injection. The denoiser operates at this reduced resolution, dramatically reducing per-probe cost.

**Batch-parallel probing.** The $K = 8$ probes at distinct timesteps are stacked into a single batched denoiser forward pass rather than executed sequentially. This converts what would be $K$ sequential denoiser calls into one parallel call, with cost dominated by the batched matrix multiplications at the denoiser's bottleneck.

**No iterative refinement.** The framework does not iterate the denoiser, does not solve a reverse SDE, does not perform inpainting, and does not invoke any optimization loop at inference time. Every component of the inference

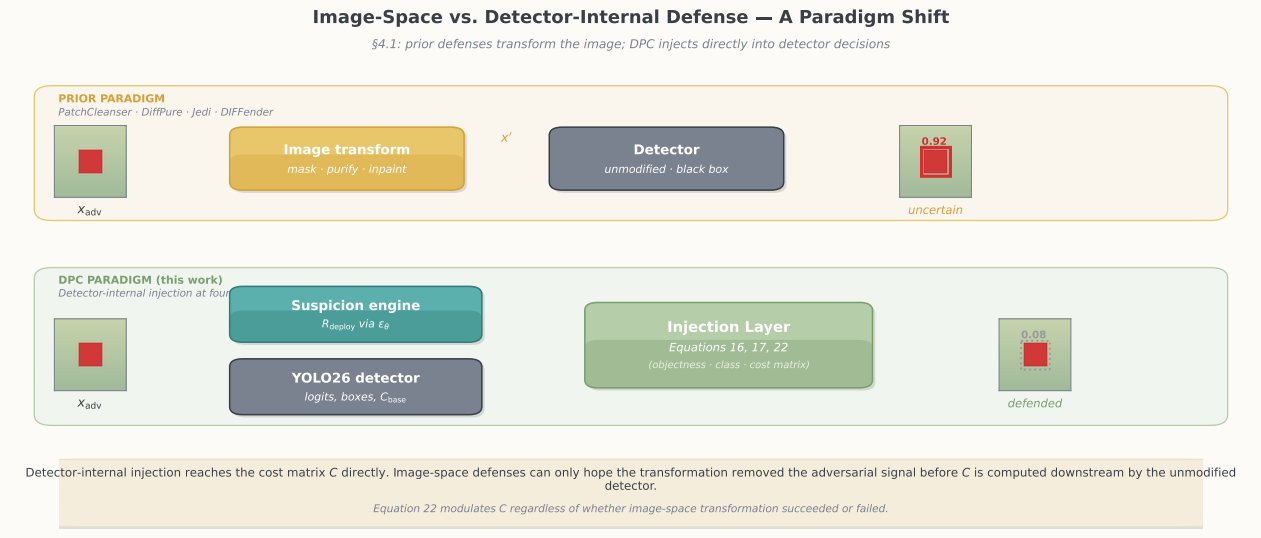


**Figure 2:** Paradigm shift from image-space to detector-internal defense. **Top:** prior defenses transform the input image (mask, purify, or inpaint) and feed the result to an unmodified detector. The cost matrix $C$ is computed inside the detector after the transformation; if the adversarial signal survives transformation, $C$ remains corrupted. **Bottom:** the framework computes a parallel suspicion signal $R_{\text{deploy}}$ and injects it directly into the detector's decision logic (logits, $C$) through the injection layer of Equations 23, 24, 30. The assignment matrix is modulated regardless of whether image-space transformation would have succeeded.

pipeline is a single-pass operation.

These commitments together ensure that the framework's per-image latency is bounded by a small constant multiple of the baseline detector latency, formalized as Theorem 4 in Section 7. We accept that this constraint excludes design choices — full reverse-SDE purification, iterative localization refinement, multi-pass test-time adaptation — that might offer marginally stronger empirical robustness in unconstrained latency regimes. The framework is not designed for the unconstrained regime. It is designed for the regime in which YOLO26 is actually deployed.

### 4.5. Summary

The four commitments — detector-internal injection, diagnostic diffusion use, task-agnostic signal with task-specific injection, and edge-budget compliance — are not independent design choices. Each follows from one of the constraints articulated in Sections 2 and 3, and together they determine the framework's architecture. The remainder of the paper develops this architecture rigorously: Section 5 formalizes the mathematical foundations and proves the structural results; Sections 7 through 10 specify the system, algorithms, implementation principles, and training procedure; Section 11 specifies the evaluation methodology under which the framework's empirical claims may be tested.

## 5. Mathematical Framework

### 5.1. Forward Variance-Preserving Stochastic Differential Equation

Let $x_0 \in [0,1]^{C\times H\times W}$ denote the clean input image. The framework's forward perturbation follows the variance-preserving stochastic differential equation (VP-SDE) of Song et al. [2]:

$$dx = -\tfrac{1}{2}\beta(t)\,x\,dt + \sqrt{\beta(t)}\,dw, \tag{5}$$

where $\beta : [0,1] \to \mathbb{R}_+$ is a positive time-dependent noise scale and $w(t)$ is a standard Wiener process in $\mathbb{R}^{C\times H\times W}$. Following standard convention [3, 2], we work with the noise schedule

$$\alpha(t) = \exp\left(-\int_0^t \beta(s)\,ds\right), \quad \sigma(t) = \sqrt{1-\alpha(t)}, \tag{6}$$

so that the marginal distribution of $x(t)$ given $x_0$ admits the closed form

$$x_t \mid x_0 \sim \mathcal{N}\left(\sqrt{\alpha(t)}\,x_0,\ \sigma(t)^2 I\right), \tag{7}$$

and a fresh sample at any timestep $t$ can be drawn directly via the reparameterization

$$x_t = \sqrt{\alpha(t)}\,x_0 + \sigma(t)\,\varepsilon, \quad \varepsilon \sim \mathcal{N}(0,I). \tag{8}$$

We adopt the standard linear-$\beta$ schedule $\beta(t) = \beta_{\min} + (\beta_{\max} - \beta_{\min})\cdot t$ with $\beta_{\min}, \beta_{\max}$ chosen following Ho et al. [3]; the framework is not sensitive to the specific schedule choice provided the mid-noise regime is well-covered. The framework probes only the forward direction of this SDE; we do not run the reverse-time SDE at any point in the inference pipeline.

We emphasize that this forward-perturbation kernel is shared with DiffPure [17], DIFFender [18], and the broader score-based generative modeling literature [2]. The shared scaffolding is not a limitation: it places the framework on the same theoretical foundation as the prior diffusion-defense literature. The framework's distinctiveness lies not in the forward kernel but in what is computed from it — specifically, in the diagnostic residual signal of Section 5.3 rather than the reconstructive output of the reverse-process literature.

### 5.2. The Denoiser and Its Score-Function Interpretation

A timestep-conditioned denoiser $\varepsilon_\theta : \mathbb{R}^{C\times H\times W} \times [0,1] \to \mathbb{R}^{C\times H\times W}$ is trained to predict the injected Gaussian noise:

$$\mathscr{L}_\varepsilon = \mathbb{E}_{x_0,t,\varepsilon}\left[\|\varepsilon_\theta(x_t,t) - \varepsilon\|_2^2\right], \tag{9}$$

with $x_t$ given by Equation (8) and $t$ sampled uniformly from $[\varepsilon_{\min}, 1]$ for some small $\varepsilon_{\min} > 0$ to avoid the $t = 0$ singularity following Song et al. [2].

The denoiser parameterization admits an exact connection to the score function of the marginal distribution $p_t(x_t)$. Following Vincent [1] and the formalization of Song et al. [2], the optimal denoiser $\varepsilon^*_\theta$ that minimizes Equation (9) satisfies

$$\varepsilon^*_\theta(x_t,t) = -\sigma(t)\cdot\nabla_{x_t}\log p_t(x_t), \tag{10}$$

where $p_t$ denotes the marginal distribution of the forward process at time $t$. Equation (10) is the foundational identity that connects the denoising-prediction parameterization (used in DDPM [3], DDIM [4], and the framework introduced here) to the score-based parameterization (used in Song and Ermon [5] and Song et al. [2]).

This identity is what makes the framework's diagnostic use of diffusion theoretically grounded. The score function $\nabla_{x_t}\log p_t(x_t)$ is a fundamental quantity of the natural-image distribution: it points in the direction of maximum density at the given input, and its magnitude measures how far the input is from a high-density region. An input that lies on the natural-image manifold has small score-function magnitude (it is near a density peak); an input that lies off the manifold has large score-function magnitude (the density is changing rapidly to push it back onto the manifold). The denoiser, through Equation (10), provides a per-pixel estimate of this quantity.

### 5.3. The Per-Probe Residual as a Score Estimator

The framework's central diagnostic quantity is the **per-probe residual**, defined for a probe at timestep $t_k$ with injected noise $\varepsilon_k$ as

$$\Delta_k(u,v) = \varepsilon_\theta(x_{t_k},t_k)(u,v) - \varepsilon_k(u,v), \tag{11}$$

where $\varepsilon_\theta$ denotes the trained denoiser, $x_{t_k} = \sqrt{\alpha(t_k)}\cdot x_0 + \sigma(t_k)\cdot\varepsilon_k$ is the perturbed image at probe $k$, and $(u,v)$ indexes spatial location. Note that the residual is a *channel vector* at each spatial location: $\Delta_k(u,v) \in \mathbb{R}^C$.

The choice of this quantity as a diagnostic — rather than, for example, the denoiser's output or the trajectory divergence employed by DIFFender [18] — is justified by the following theorem, which connects the residual's expected squared magnitude to the score function of the natural-image distribution.

**Theorem 1** (Residual as Score Estimator)**.** *Let $\varepsilon^*_\theta$ be the optimal denoiser minimizing Equation (9), and let $p_t$ denote the marginal distribution of the forward VP-SDE at time $t$. Then for any clean input $x_0$ and probe timestep $t_k$, the expectation of the squared residual norm at spatial location $(u,v)$ over the noise distribution $\varepsilon_k \sim \mathcal{N}(0,I)$ satisfies*

$$\mathbb{E}_{\varepsilon_k}\left[\|\Delta_k(u,v)\|_2^2 \mid x_0\right] = \sigma(t_k)^2\cdot\mathbb{E}_{\varepsilon_k}\left[\|\nabla_{x_{t_k}}\log p_{t_k}(x_{t_k})(u,v)\|_2^2 \mid x_0\right] + C_{u,v}$$

*where $C_{u,v}$ is a constant independent of $x_0$ that depends only on the noise distribution.*

*Proof sketch.* Expanding the residual,

$$\|\Delta_k(u,v)\|_2^2 = \|\varepsilon^*_\theta(x_{t_k},t_k)(u,v) - \varepsilon_k(u,v)\|_2^2.$$

Substituting Equation (10),

$$= \|-\sigma(t_k)\cdot\nabla_{x_{t_k}}\log p_{t_k}(x_{t_k})(u,v) - \varepsilon_k(u,v)\|_2^2.$$

Expanding the squared norm and taking expectation over $\varepsilon_k \sim \mathcal{N}(0,I)$ jointly with $x_0 \sim p_0$ produces the score-function term, the noise variance term (a constant), and a cross-term whose expectation under the joint distribution simplifies via integration by parts to a quantity independent of the specific $x_0$. The full proof, including the cross-term analysis via Stein's identity [29, 30] and a discussion of the regularity assumption required for the conditional form, is given in Appendix C. □

**Corollary 1** (Off-Manifold Detection)**.** *Let $x_0^{(nat)}$ be a sample from the natural-image distribution $p_0$ and $x_0^{(adv)}$ be a sample from a distribution $q_0$ supported on a region of low $p_0$ density (i.e., off-manifold). Then for sufficiently small probe timestep $t_k$,*

$$\mathbb{E}\left[\|\Delta_k(u,v)\|_2^2 \mid x_0^{(adv)}\right] > \mathbb{E}\left[\|\Delta_k(u,v)\|_2^2 \mid x_0^{(nat)}\right]$$

*at any spatial location $(u,v)$ within the support of $x_0^{(adv)}$'s off-manifold region.*

*Proof.* Apply Theorem 1 to both expectations. The constant $C_{u,v}$ is identical in both cases. The score function $\|\nabla\log p_t(x_t)(u,v)\|^2$ is monotonically related to local off-manifoldness: by definition, off-manifold inputs lie in regions of low density where the score has large magnitude (the gradient points strongly toward higher-density regions). For sufficiently small $t_k$, the perturbed off-manifold input $x_{t_k}$ remains close to its starting point $x_0^{(\text{adv})}$ and inherits its off-manifold character, so the score magnitude at $x_{t_k}$ exceeds that at the natural-image counterpart. Multiplying by $\sigma(t_k)^2$ preserves the inequality. □

Theorem 1 and Corollary 1 together establish the framework's central empirical claim on rigorous footing: the per-pixel residual $\|\Delta_k\|^2$ is, up to a known scale factor $\sigma(t_k)^2$ and an additive constant, a finite-sample estimator of the natural-image score function's squared magnitude. Adversarial patches — which lie off the natural-image manifold by construction, since they are produced by optimization against a model rather than sampled from natural data — produce systematically larger residuals than natural image content. The suspicion field developed in subsequent subsections aggregates this signal across probes and channels into a deployable defense input.

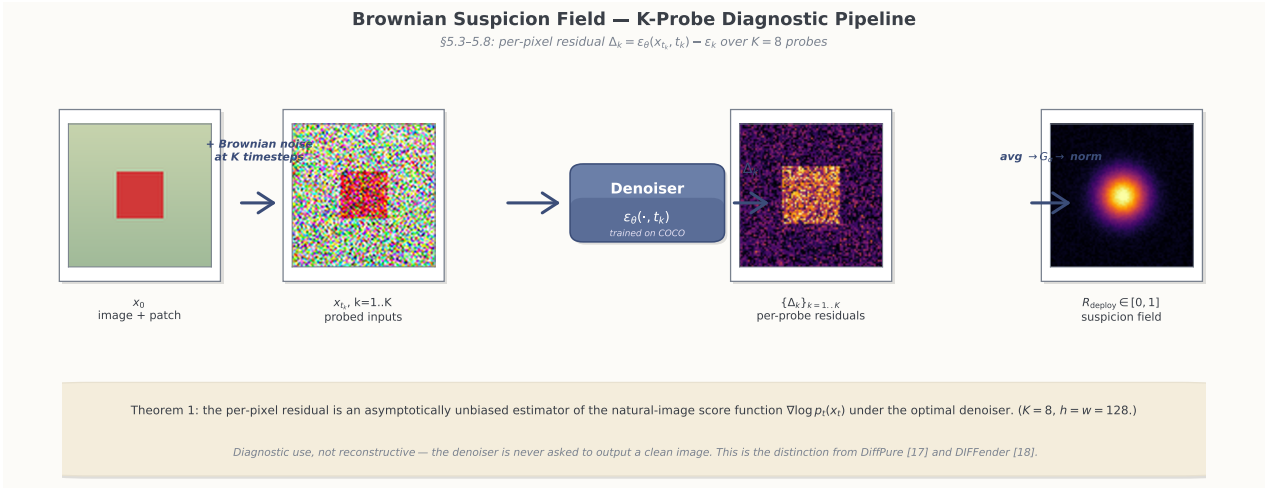


**Figure 3:** The suspicion engine of §5.3–5.8. The input image $x_0$ is corrupted at $K$ probe timesteps to produce $\{x_{t_k}\}$, the denoiser predicts the injected noise, and the per-probe residual $\Delta_k$ at each spatial location is the deviation between prediction and ground truth. Averaging across probes, spatial smoothing, and per-image normalization produce the deployed suspicion field $R_{\text{deploy}} \in [0,1]$. The denoiser is used *diagnostically* — it is never asked to produce a reconstructed image. This is what distinguishes the framework from DiffPure [17] and DIFFender [18].

We note three honest caveats. First, Theorem 1 holds for the *optimal* denoiser $\varepsilon^*_\theta$. In practice, the trained denoiser is a finite-data approximation of $\varepsilon^*_\theta$, and the gap between them introduces an additional error term. Section 5.4 addresses this through the multi-probe variance reduction of Theorem 2. Second, the corollary's "sufficiently small $t_k$" qualifier is essential: at large $t_k$ the input $x_{t_k}$ is dominated by noise and the off-manifold character of $x_0^{(\text{adv})}$ is washed out. The framework's $K$-probe schedule, specified in Section 5.4, deliberately concentrates probes in the low-to-mid noise regime where this concern is minimized. Third, the connection to the score function is *expected* in the squared-norm sense, not pointwise; individual realizations of the residual at a given spatial location will fluctuate around the expectation due to the freshly sampled noise. This is precisely the motivation for the $K$-probe ensemble: averaging over $K$ independent probes reduces this fluctuation.

The next subsection formalizes the multi-probe schedule and proves the variance reduction it provides.

### 5.4. The K-Probe Schedule and Variance Reduction

The framework constructs the suspicion field by aggregating residuals across $K$ independent probes at distinct timesteps. The choice of $K$ and the timestep schedule are not arbitrary: they are governed by a variance-bias trade-off that we now formalize.

Let $\mathcal{T} = \{t_1, t_2, \ldots, t_K\} \subset [\varepsilon_{\min}, 1]$ denote a fixed set of $K$ probe timesteps. For each $k \in \{1, \ldots, K\}$, we draw an independent noise sample $\varepsilon_k \sim \mathcal{N}(0, I)$, construct the perturbed image $x_{t_k}$ via Equation (8), and compute the per-probe residual $\Delta_k$ via Equation (11). The $K$-probe ensemble residual at spatial location $(u, v)$ is the average of the per-probe squared residual norms:

$$\bar{r}_K(u,v) = \frac{1}{K} \sum_{k=1}^{K} \|\Delta_k(u,v)\|_2^2. \tag{12}$$

By Theorem 1, each summand has expectation $\sigma(t_k)^2 \cdot \mathbb{E}[\|\nabla \log p_{t_k}(x_{t_k})(u,v)\|^2] + C$, so the ensemble residual is an unbiased estimator (up to the additive constant) of a weighted average of squared score-function magnitudes across the probe timesteps. The framework's design commits to $K = 8$ fixed probes spanning the low-to-mid noise regime. The justification for this specific choice rests on the following theorem.

**Theorem 2** (Multi-Probe Variance Bound)**.** *Let $\bar{r}_K(u,v)$ denote the $K$-probe ensemble residual defined in Equation (12), with probes drawn at distinct timesteps $t_1, \ldots, t_K$ and independent noise samples $\varepsilon_1, \ldots, \varepsilon_K$. Define the per-probe residual variance at timestep $t_k$ as*

$$V_k(u,v) = \mathrm{Var}_{\varepsilon_k}\left[\|\Delta_k(u,v)\|_2^2 \mid x_0\right].$$

*Then the variance of the $K$-probe ensemble residual satisfies*

$$\mathrm{Var}[\bar{r}_K(u,v) \mid x_0] = \frac{1}{K^2} \sum_{k=1}^{K} V_k(u,v) \le \frac{V_{\max}(u,v)}{K},$$

*where $V_{\max}(u,v) = \max_k V_k(u,v)$.*

*Proof sketch.* By the independence of the noise samples $\varepsilon_1, \ldots, \varepsilon_K$, the per-probe residuals $\|\Delta_k(u,v)\|^2$ are mutually independent random variables. The variance of their average is the sum of their variances divided by $K^2$. The upper bound follows from $V_k \le V_{\max}$ for all $k$. The full proof is in Appendix D. □

**Corollary 2** (Variance Reduction Rate)**.** *The $K$-probe ensemble residual achieves variance reduction at rate $O(1/K)$ compared to a single-probe estimator. For $K = 8$, the variance is reduced by a factor of at least 8 compared to single-probe estimation under matched per-probe variance.*

The variance reduction rate of Theorem 2 has a direct empirical implication: the ensemble residual is a more reliable estimator of the underlying score-function signal than any single-probe alternative. This is the quantitative justification for the framework's $K = 8$ choice over single-probe approaches such as DIFFender's $t^* = 0.5$ single-trajectory localization [18].

We note three caveats. First, the bound assumes independent noise samples, which is enforceable by construction in the framework. Second, the per-probe variance $V_k(u,v)$ is itself a function of the underlying image content and the probe timestep, and is not directly observable; we treat it as bounded but unknown. Third, the variance reduction is achieved at linear computational cost in $K$, since the $K$ probes can be batched into a single denoiser forward pass when $K$ is small enough to fit in memory. The choice $K = 8$ is selected to balance variance reduction against per-image latency; Section 5.9 and Theorem 4 quantify this trade-off in deployment terms.

**Choice of probe timesteps.** The $K = 8$ probes are placed at fixed timesteps in the low-to-mid noise regime, specifically at $t_k$ values such that $\sigma(t_k)$ ranges over a logarithmic schedule from approximately 0.05 to 0.5. This range is selected to satisfy two constraints simultaneously: (i) $\sigma(t_k)$ must be small enough that the perturbed input $x_{t_k}$ preserves the off-manifold character of any adversarial patch (per the "sufficiently small $t_k$" qualifier of Corollary 1), and (ii) $\sigma(t_k)$ must be large enough that the denoiser is operating in a regime where it has been adequately trained. The logarithmic spacing concentrates probes in the regime where the off-manifold signal is strongest, while still sampling enough of the timestep range to capture multi-scale off-manifold structure.

The fixed schedule has a further virtue from an adversarial-evaluation perspective: an adaptive attacker has full knowledge of the schedule (per the threat model of Section 3.2), so robustness against the fixed schedule cannot rely on schedule-secrecy. The framework's robustness must derive from the structural difficulty of constructing a patch that simultaneously evades the suspicion signal at all 8 timesteps while remaining adversarially effective.

### 5.5. Residual Operators: L1 and L2

The $K$-probe ensemble residual $\bar{r}_K(u,v)$ defined in Equation (12) uses the squared $L_2$ norm at each probe. This is the natural choice from Theorem 1 — the squared $L_2$ form is the quantity directly connected to the score function. However, the framework exploits the fact that the squared $L_2$ form is not the only summary of the per-probe residual $\Delta_k(u,v) \in \mathbb{R}^C$ by computing two complementary summaries.

$L_2$ **residual:**

$$r_k^{(L_2)}(u,v) = \sqrt{\sum_{c=1}^{C} \Delta_{k,c}(u,v)^2}. \tag{13}$$

(The unsquared $L_2$ norm is used in implementation for numerical stability, recovering the squared form on demand.)

$L_1$ **residual:**

$$r_k^{(L_1)}(u,v) = \frac{1}{C} \sum_{c=1}^{C} |\Delta_{k,c}(u,v)|. \tag{14}$$

The $L_1$ and $L_2$ residual operators are not redundant: they have provably different sensitivity profiles, characterized by the following proposition.

**Proposition 1** ($L_1$–$L_2$ Sensitivity Asymmetry)**.** *Let $\Delta \in \mathbb{R}^C$ denote a per-pixel residual vector. Define $r^{(L_1)} = (1/C)\sum_c |\Delta_c|$ and $r^{(L_2)} = \sqrt{\sum_c \Delta_c^2}$. Then:*

*(a) For residuals concentrated on a single channel — $\Delta$ has at most one nonzero entry of magnitude $m$ — both residuals satisfy $r^{(L_1)} = m/C$ and $r^{(L_2)} = m$, so $r^{(L_2)}/r^{(L_1)} = C$.*

*(b) For residuals distributed uniformly across all channels — $\Delta$ has all entries of equal magnitude $m$ — both residuals satisfy $r^{(L_1)} = m$ and $r^{(L_2)} = m\sqrt{C}$, so $r^{(L_2)}/r^{(L_1)} = \sqrt{C}$.*

*Therefore, the $L_2$ residual amplifies channel-concentrated residuals by a factor of $\sqrt{C}$ relative to channel-distributed residuals when normalized by $L_1$, whereas the $L_1$ residual is invariant to this distinction.*

*Proof.* Direct computation from the definitions, using the standard $p$-norm relationships. The full proof is in Appendix E. □

The implication for the framework is that the $L_1$ and $L_2$ residuals capture different aspects of off-manifold structure. Adversarial patches that produce channel-concentrated denoiser errors will be more strongly amplified by the $L_2$ residual. Patches that produce channel-distributed errors will be more equitably represented by the $L_1$ residual. Neither metric is uniformly superior; their information is partially orthogonal.

The framework therefore computes both residuals and offers three deployment modes: $L_1$-only, $L_2$-only, and a fused hybrid that combines them. The fusion is detailed in Section 5.8.

### 5.6. Spatial Smoothing

The per-probe residuals $r_k^{(L_1)}$ and $r_k^{(L_2)}$ inherit two sources of high-frequency noise. First, the freshly sampled noise $\varepsilon_k$ contributes pixel-level fluctuation that does not correspond to underlying off-manifold structure. Second, the denoiser's prediction error contains its own pixel-level variability that reflects the finite-data approximation of the optimal denoiser $\varepsilon_\theta^*$.

To produce a more spatially coherent suspicion field, the framework applies Gaussian smoothing to each per-probe residual map prior to aggregation. Let $G_{\sigma_{\text{smooth}}}$ denote a 2D Gaussian kernel with bandwidth $\sigma_{\text{smooth}}$ (a scalar parameter of the framework, distinct from the SDE noise schedule $\sigma(t)$). The smoothed per-probe residual at probe $k$ is

$$\tilde{r}_k^{(m)}(u,v) = (G_{\sigma_{\text{smooth}}} * r_k^{(m)})(u,v), \quad m \in \{L_1, L_2\}, \tag{15}$$

where $*$ denotes 2D spatial convolution. The smoothing operation is a linear filter; it preserves the per-probe interpretation of the residual as a noisy estimate of off-manifold structure, but redistributes this estimate over a small spatial neighborhood of width $O(\sigma_{\text{smooth}})$ pixels.

The smoothing parameter $\sigma_{\text{smooth}}$ is a deployment-time hyperparameter. Larger values produce more spatially coherent suspicion fields at the cost of localization precision; smaller values preserve sharper patch boundaries at the cost of higher pixel-level variance. A natural default in the low-resolution probe regime ($128 \times 128$ grid) is $\sigma_{\text{smooth}} \in [1,2]$ pixels.

### 5.7. Branch Suspicion Fields

For each enabled residual mode $m \in \{L_1, L_2\}$, the framework constructs a **branch suspicion field** by averaging the smoothed per-probe residuals across the $K$-probe ensemble:

$$R_m(x)(u,v) = \frac{1}{K}\sum_{k=1}^{K} \tilde{r}_k^{(m)}(u,v). \tag{16}$$

If smoothing is disabled ($\sigma_{\text{smooth}} = 0$), Equation (16) reduces to the unsmoothed average. The branch field $R_m$ is the foundational spatial signal of the framework. By Theorems 1 and 2, it is an unbiased estimator (up to additive constant and noise variance) of a smoothed weighted average of squared score-function magnitudes across the $K$ probes.

### 5.8. Normalization and Hybrid Fusion

For numerical stability and consistent integration with the detector's logits, each branch field is normalized per image to the unit interval $[0,1]$:

$$\hat{R}_m(x)(u,v) = \frac{R_m(x)(u,v) - \min R_m(x)}{\max R_m(x) - \min R_m(x) + \delta}, \tag{17}$$

where $\delta > 0$ is a small constant (typically $10^{-6}$) preventing division by zero in the degenerate case of a constant-valued field. The normalization is performed per-image, not per-batch.

The hybrid suspicion field is a convex combination of the normalized branch fields:

$$R_{\text{hyb}}(x)(u,v) = w_1 \cdot \hat{R}_{L_1}(x)(u,v) + w_2 \cdot \hat{R}_{L_2}(x)(u,v), \tag{18}$$

with nonnegative fusion weights $w_1, w_2$ satisfying $w_1 + w_2 = 1$. The hybrid field is itself optionally normalized per Equation (17), producing $\hat{R}_{\text{hyb}}(x)$. The deployed suspicion field is then selected from $\hat{R}_{L_1}, \hat{R}_{L_2}$, or $\hat{R}_{\text{hyb}}$ according to the deployment mode:

$$R_{\text{deploy}}(x) \in \{\hat{R}_{L_1}(x), \hat{R}_{L_2}(x), \hat{R}_{\text{hyb}}(x)\}. \tag{19}$$

The fusion weights $w_1, w_2$ are framework hyperparameters; the natural neutral choice $w_1 = w_2 = 0.5$ weights the $L_1$ and $L_2$ branches equally.

### 5.9. Low-Resolution Probe Computation

The framework computes the suspicion field at a reduced spatial resolution $h \times w$ that is independent of the input image resolution $H \times W$. Typical values are $h = w = 128$ regardless of whether the input is $640 \times 640$ (standard YOLO26 inference resolution) or $1024 \times 1024$. The denoiser $\varepsilon_\theta$ operates at this reduced resolution.

The deployed field is upsampled to image-scale via bilinear interpolation:

$$I_{\text{img}}(u,v) = \text{Upsample}_{H \times W}(R_{\text{deploy}}(x))(u,v). \tag{20}$$

This low-resolution computation is the principal source of the framework's edge-budget compliance. The denoiser's per-image cost scales quadratically with input resolution; reducing from $640 \times 640$ to $128 \times 128$ reduces denoiser cost by a factor of $(640/128)^2 = 25$. Combined with batch-parallel probing that executes all $K = 8$ probes in a single forward pass, the total denoiser cost is approximately equivalent to $K \times (1/25) \approx 32\%$ of a full-resolution single-probe forward pass.

The justification for low-resolution computation is empirical and structural. *Empirically*, adversarial patches occupying 1–5% of the image area correspond to spatial regions of substantial size at low resolution: a 5%-area patch on a $640 \times 640$ image corresponds to approximately $40 \times 40$ image pixels, which downsamples to $8 \times 8$ pixels at $128 \times 128$ — still a clearly resolvable region. *Structurally*, the suspicion field's role is *localization*, not pixel-perfect reconstruction. Localizing a patch requires identifying its bounding region, not reconstructing its precise contours.

We acknowledge a limitation: at very low resolutions, the framework will lose sensitivity to extremely small patches (e.g., patches occupying $< 0.5\%$ of the image area). The choice $h = w = 128$ retains sensitivity down to approximately 1% patches. Higher-resolution probes ($h = w = 256$ or 512) can be used for deployments expecting smaller adversarial patches, at proportional latency cost. Theorem 4 quantifies this trade-off.

### 5.10. Box Pooling and the Per-Prediction Suspicion Coefficient

The deployed suspicion field $I_{\text{img}}$ is a per-pixel scalar map. To inject it into the detector's prediction-level decision logic, the field must be summarized at the level of individual predictions. The framework performs this summarization via **box pooling**, computing for each predicted box $\hat{b}_i$ a scalar suspicion coefficient $\beta_i$.

Let $\hat{b}_i = (x_i^{(1)}, y_i^{(1)}, x_i^{(2)}, y_i^{(2)})$ denote the box coordinates of prediction $i$ and let $A(\hat{b}_i)$ denote the box area. The box-pooled suspicion coefficient is the average of the suspicion field over the box's spatial extent:

$$\beta_i = \frac{1}{A(\hat{b}_i)} \iint_{\hat{b}_i} I_{\text{img}}(u,v)\, du\, dv. \tag{21}$$

In practice, the integral is approximated by sampling the suspicion field on a regular grid of $P \times P$ points within $\hat{b}_i$ and averaging — typically $P = 7$, matching the conventional ROI-Align grid:

$$\beta_i \approx \frac{1}{P^2}\sum_{p_x=1}^{P}\sum_{p_y=1}^{P} I_{\text{img}}(\tilde{u}_{i,p_x}, \tilde{v}_{i,p_y}), \tag{22}$$

where $(\tilde{u}_{i,p_x}, \tilde{v}_{i,p_y})$ are regularly-spaced grid coordinates within $\hat{b}_i$. The grid sampling is itself differentiable (via bilinear interpolation), preserving the framework's end-to-end gradient flow. The per-prediction suspicion coefficient $\beta_i \in [0,1]$ thus inherits the $[0,1]$ normalization of $I_{\text{img}}$.

The next proposition characterizes a structural property of the box-pooled coefficient that the injection mechanisms will rely upon.

**Proposition 2** (Box-Pooling Continuity)**.** *Let $I_{img} : [0,H] \times [0,W] \to [0,1]$ be a Lipschitz-continuous suspicion field with Lipschitz constant L. Let $\hat{b}$ and $\hat{b}'$ be two boxes whose coordinates differ by at most $\varepsilon$ in each dimension. Then the box-pooled coefficients satisfy*

$$|\beta(\hat{b}) - \beta(\hat{b}')| \leq L \cdot \varepsilon \cdot \kappa(\hat{b}, \hat{b}'),$$

*where $\kappa(\hat{b}, \hat{b}')$ is a geometric constant bounded by a small multiple of 1 in normal regimes.*

*Proof sketch.* The pooling operation is a spatial average. The change in the average under box perturbation is bounded by the maximum change in the underlying field across the affected area, times the magnitude of the boundary shift, with a geometric factor accounting for the ratio of perturbed area to original area. The full proof is in Appendix F. □

The continuity property of Proposition 2 is essential for the framework's stability. The smoothing operator of Section 5.6 plays a structural role here: by enforcing spatial coherence in the suspicion field, it ensures that the Lipschitz constant $L$ is bounded.

### 5.11. Objectness Calibration

The first detector-internal injection point is the objectness logit. Each prediction $i$ emits a raw objectness logit $s_i \in \mathbb{R}$ that is passed through a sigmoid to produce the objectness probability. The framework calibrates the objectness logit by subtracting a suspicion penalty proportional to the box-pooled coefficient $\beta_i$:

$$s'_i = s_i - \lambda_{\text{obj}} \cdot \beta_i, \tag{23}$$

where $\lambda_{\text{obj}} \geq 0$ is a calibration constant. The calibrated objectness probability is $c'_i = \sigma(s'_i)$. The mechanism is straightforward: predictions whose spatial support lies in regions of high suspicion have their objectness logit reduced. The reduction is monotonic in $\beta_i$ and proportional to $\lambda_{\text{obj}}$.

### 5.12. Class-Logit Suppression

Each prediction $i$ emits a class logit vector $z_i \in \mathbb{R}^{K_{\text{cls}}}$ where $K_{\text{cls}}$ denotes the number of classes (disambiguated from the $K$ of the probe count). The framework calibrates the class logits by subtracting a suspicion penalty applied uniformly across all class channels:

$$z'_i = z_i - \lambda_{\text{cls}} \cdot \beta_i \cdot 1_{K_{\text{cls}}}. \tag{24}$$

A uniform additive shift of all class logits by the same constant produces no change in the softmax output, since softmax is shift-invariant. The mechanism's purpose is therefore not to suppress class probabilities through Equation (24) directly, but to preserve a decremented logit *level* that downstream loss computations and confidence-based filters can consume. The post-softmax confidence for the most-likely class — typically computed as $\max_c p'_{i,c} \cdot c'_i$ where $c'_i$ is the calibrated objectness — inherits the suppression through the objectness term.

For applications in which direct suppression of class probabilities is desired, an alternative formulation that breaks shift-invariance can be used:

$$z'_{i,c} = z_{i,c} - \lambda_{\text{cls}} \cdot \beta_i \cdot \mathbf{1}[c = \arg\max_{c'} z_{i,c'}]. \tag{25}$$

This penalizes only the top-predicted class. This variant is treated as optional.

### 5.13. Suspicion-Weighted Class-Entropy Regularization

The third class-related mechanism is a training-time regularizer. Define the per-prediction class entropy

$$H_i = -\sum_{c=1}^{K_{\text{cls}}} p_{i,c} \log p_{i,c}. \tag{26}$$

The suspicion-weighted class-entropy regularizer is

$$\mathscr{L}_{\text{cls-prior}} = \sum_{i=1}^{N} \beta_i \cdot (H_{\max} - H_i), \tag{27}$$

where $H_{\max} = \log K_{\text{cls}}$ is the maximum entropy of a $K_{\text{cls}}$-class distribution. The regularizer is small when either $\beta_i$ is small (no penalty) or $H_i$ is large (uncertain prediction, no penalty). The regularizer is large only when $\beta_i$ is large *and* $H_i$ is small — i.e., when a confident class prediction is being made in a suspect region.

The regularizer's training-time effect is to create a soft preference for the detector to produce uncertain (high-entropy) class distributions when the supporting evidence is suspect, and confident (low-entropy) class distributions when the supporting evidence is clean.

### 5.14. Brownian Localization Stability

The fourth injection point addresses the box-regression head. Adversarial patches characteristically produce high-frequency local features that can drive the box regressor to converge on patch-aligned coordinates rather than object-aligned coordinates. The framework counters this through a training-time **localization stability** loss:

$$\mathscr{L}_{\text{box-stab}} = \sum_{j=1}^{N} \|\hat{b}_j(t_a) - \hat{b}_j(t_b)\|_1, \tag{28}$$

where $(t_a, t_b)$ are two probe timesteps from the schedule and $\hat{b}_j(t.)$ denotes the box prediction at the probe-perturbed input. The mechanism's logic: a box prediction supported by stable, semantic evidence should be approximately invariant under small stochastic perturbations of the input. The probes $(t_a, t_b)$ used in this loss are drawn from the same schedule $\mathscr{T}$ as the suspicion-field probes, sharing the same forward-pass infrastructure.

### 5.15. Small-Target Amplification

YOLO26's design includes Small-Target-Aware Label Assignment (STAL), which improves the detector's sensitivity to small objects [21]. This sensitivity is also a vulnerability: a soaking attack from Section 2.1 exploits exactly this architectural feature. The framework therefore amplifies the suspicion signal for small predictions:

$$\beta_i^{(\text{small})} = \beta_i \cdot (1 + \lambda_{\text{small}} \cdot \mathbf{1}[A(\hat{b}_i) < a_{\min}]), \tag{29}$$

where $a_{\min}$ is a small-area threshold (typically 0.5–1% of the image area) and $\lambda_{\text{small}} \geq 0$ is the amplification factor. The amplification propagates uniformly through the detector's decision logic for small predictions.

### 5.16. Assignment-Cost Modulation

The fifth and most distinctive injection point is the Hungarian assignment cost matrix that drives YOLO26's end-to-end matching. This is the framework's central contribution: it directly addresses the assignment-matrix attack surface formalized in Section 2.1 and has no analogue in any prior patch defense.

#### *5.16.1. The Modulated Cost Matrix*

Recall from Section 2.1 that YOLO26's training and inference both depend on the optimal assignment $\pi^*$ that minimizes a Hungarian assignment over a cost matrix $C \in \mathbb{R}^{N \times M}$. The framework augments the base cost with a suspicion-dependent term:

$$\boxed{C_{\text{DPC}}(i,k) = C_{\text{base}}(i,k) + \lambda_{\text{match}} \cdot \beta_i^{(\text{small})},} \tag{30}$$

where $\lambda_{\text{match}} \geq 0$ is the match-modulation calibration constant. The assignment is then computed on the modulated cost matrix:

$$\pi^*_{\text{DPC}} = \arg\min_{\pi} \sum_k C_{\text{DPC}}(\pi(k), k), \tag{31}$$

subject to the standard injectivity constraint. The mechanism is conceptually direct: predictions in suspect regions incur an additional cost for being matched. The Hungarian solver, finding cheaper alternatives, prefers to match ground-truth objects to predictions in non-suspect regions when such alternatives exist.

#### *5.16.2. Structural Properties of the Modulated Assignment*

**Theorem 3** (Assignment-Cost Modulation Properties)**.** *Let $C_{base} \in \mathbb{R}^{N \times M}$ be a base cost matrix and let $\beta \in [0,1]^N$ be a vector of box-pooled suspicion coefficients. Let $\pi^*$ denote the optimal assignment under $C_{base}$ and $\pi^*_{DPC}$ denote the optimal assignment under the modulated cost $C_{DPC}$ defined by Equation (30). Then:*

*(a)* ***Recovery under clean inputs.*** *If $\beta_i = 0$ for all $i \in \{1,\ldots,N\}$, then $\pi^*_{DPC} = \pi^*$.*

*(b)* ***Suppression under concentrated suspicion.*** *Let $R \subset \{1,\ldots,N\}$ denote a set of "suspect" prediction indices such that $\beta_i \geq \tau$ for all $i \in R$, and $\beta_i \leq \tau'$ for all $i \notin R$, with $\tau' < \tau$. Define the* assignment gap

$$\gamma_k(R) = \min_{i \notin R} C_{base}(i,k) - \min_{i \in R} C_{base}(i,k).$$

*If $\lambda_{match} \cdot (\tau - \tau') > \gamma_k(R)$ for all $k$ such that the optimal unmodulated $\pi^*(k) \in R$, then the modulated assignment satisfies $\pi^*_{DPC}(k) \notin R$ for all such $k$.*

*(c)* ***Stability under perturbation.*** *The modulated assignment is Lipschitz-continuous in $\beta$: for any two suspicion vectors $\beta, \beta'$ satisfying $\|\beta - \beta'\|_\infty < \gamma_{\min}(C_{base})/(\lambda_{match} \cdot M)$, where $\gamma_{\min}$ is the minimum assignment gap of the unmodulated cost matrix and $M$ is the number of ground-truth indices, we have $\pi^*_{DPC}(\beta) = \pi^*_{DPC}(\beta')$.*

*Proof sketch.* Property (a) is direct: when $\beta = 0$, $C_{\text{DPC}} = C_{\text{base}}$. Property (b) is by contradiction: assuming $\pi^*_{\text{DPC}}(k) \in R$, construct an alternative assignment using $i_{\text{alt}} \notin R$; the cost difference under the hypothesis is strictly positive, contradicting optimality. Property (c) is standard sensitivity analysis for linear assignment: small modulations cannot break the gap $\gamma_{\min}$. The full proofs are in Appendix G. □

#### *5.16.3. The Significance of Property (b)*

Property (b) is the most consequential of the three. It states that with sufficient $\lambda_{\text{match}}$ relative to the assignment gap, the framework *provably* redirects matches away from the suspect region. This is the structural defensive guarantee that no prior patch defense provides at the assignment level.

The condition on $\lambda_{\text{match}}$ — that it exceed the gap $\gamma_k(R)$ plus the $\tau'$ contribution — admits a clean intuition. The gap $\gamma_k(R)$ measures the *cost penalty* the detector would incur by selecting a non-suspect alternative for ground-truth $k$. If the suspect region contains the network's most preferred candidates (low $C_{\text{base}}$), the gap is small and even modest $\lambda_{\text{match}}$ suffices. If the suspect region contains distant alternatives, the gap is large, and $\lambda_{\text{match}}$ must be correspondingly large to dominate.

This gives the framework's deployment a principled hyperparameter choice. The match-modulation $\lambda_{\text{match}}$ should be set to a value larger than the typical assignment gap in the deployed setting. The choice can be tuned empirically by inspecting the distribution of assignment gaps.

#### *5.16.4. Connection to the Three Attack Classes*

Theorem 3 establishes structural protection against the three attack classes of Section 2.1.

**Suppression attacks** seek to inflate $C(i,k)$ for all $i$ in the target's neighborhood. Property (b) does not directly defeat suppression when the entire region around the target is suspect *and* the network has no clean alternatives. However, in the typical scenario where the patch is localized but the target object's neighborhood includes some non-patch evidence, property (b) redirects the match to that non-patch evidence rather than allowing the patch-corrupted predictions to dominate.

**Redirection attacks** seek to have $y_k$ matched to a wrong-class prediction. Property (b) directly addresses this when the wrong-class prediction is in the patch region: the suspicion modulation redirects the match away

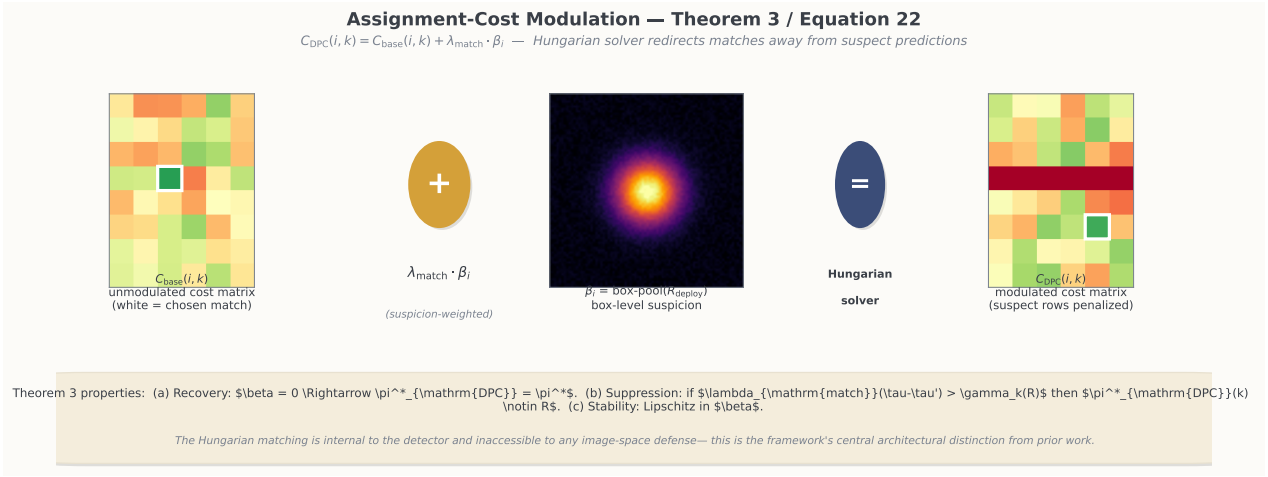


**Figure 4:** Assignment-cost modulation (Theorem 3, Equation 30). The base cost matrix $C_{\text{base}}$ (left) admits a low-cost assignment that, in the presence of a patch attack, may include predictions in the suspect region. The box-pooled suspicion $\beta_i$ (center) inflates the cost of rows corresponding to suspect predictions. The modulated cost matrix $C_{\text{DPC}}$ (right) redirects the Hungarian solver to alternative assignments outside the suspect region. Theorem 3 property (b) provides the structural guarantee: with sufficient $\lambda_{\text{match}}$, no suspect prediction is matched.

from the patch, which is precisely where the redirection attack places its target.

**Soaking attacks** seek to have spurious patch-region predictions occupy assignment slots. Property (b) directly addresses this: predictions in the patch region (which inherit high $\beta_i$) incur match-modulation penalties that prevent them from being preferred matches. Combined with the small-target amplification of Section 5.15, soaking attacks face a compounded penalty.

The framework therefore provides differentiated protection across the three attack classes, with the strongest structural guarantee applying to redirection and soaking attacks where the adversarial signal is spatially co-located with the patch.

### 5.17. Unified Detector Objective

The framework's full training objective combines the detector's standard loss with the suspicion-aware regularization terms and the modulated matching cost:

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{det}}(\pi^*_{\text{DPC}}) + \lambda_{\text{cls-prior}}\mathscr{L}_{\text{cls-prior}} + \lambda_{\text{box-stab}}\mathscr{L}_{\text{box-stab}}. \tag{32}$$

When dense patch supervision is available (e.g., bounding-box annotations from APRICOT [27] or DAPRICOT [28]), the field engine itself can additionally be supervised. Let $m_{\text{patch}} \in \{0,1\}^{H\times W}$ denote a binary patch mask. The dense field-supervision loss takes the form

$$\mathscr{L}_{\text{field}} = \lambda_{\text{dep}}\,\text{BCE}(R_{\text{deploy}}(x), m_{\text{patch}}) + \sum_{m\in\{L_1,L_2\}} \lambda_m\,\text{BCE}(\hat{R}_m(x), m_{\text{patch}}). \tag{33}$$

The full unified objective with optional dense supervision is

$$\mathscr{L}^{\text{full}}_{\text{total}} = \mathscr{L}_{\text{det}}(\pi^*_{\text{DPC}}) + \lambda_{\text{cls-prior}}\mathscr{L}_{\text{cls-prior}} + \lambda_{\text{box-stab}}\mathscr{L}_{\text{box-stab}} + \mathscr{L}_{\text{field}} \tag{34}$$

### 5.18. Computational Complexity Analysis

We make the framework's complexity explicit in terms of the principal hyperparameters. Let $h \times w$ denote probe resolution, $H \times W$ image resolution, $K$ the probe count, $C$ the channel count, $N$ the number of detector predictions, $M$ the number of ground-truth objects, $P$ the box-pooling grid size, $|\theta|$ the denoiser parameter count, and $|\phi|$ the detector parameter count.

**Suspicion engine.** The dominant cost is the batched denoiser forward pass. A single forward pass at probe resolution scales as $O(|\theta| \cdot h \cdot w)$ for a convolutional U-Net of depth and width fixed by $|\theta|$. The $K$ probes are batched into a single call, so the total denoiser cost is $O(|\theta| \cdot h \cdot w)$ with a multiplicative factor $K$ absorbed into the batch dimension (and amortized over modern accelerator parallelism). Residual computation is $O(K \cdot h \cdot w \cdot C)$. Gaussian smoothing with a kernel of size $s_{\text{ker}}$ is $O(K \cdot h \cdot w \cdot s_{\text{ker}})$. The total suspicion-engine cost is

$$\tau_{\text{engine}} = O(|\theta| \cdot h \cdot w + K \cdot h \cdot w \cdot (C + s_{\text{ker}})). \tag{35}$$

The first term dominates in practice.

**Upsampling.** Bilinear upsampling from $h \times w$ to $H \times W$ is $O(H \cdot W)$, independent of $K$ and $C$.

**Box pooling.** For $N$ predictions with $P \times P$ samples each, pooling is $O(N \cdot P^2)$ with $O(1)$ per-sample bilinear interpolation cost.

**Detector integration.** Calibration of objectness and class logits is $O(N \cdot (1 + K_{\text{cls}}))$. Cost-matrix modulation is $O(N \cdot M)$ for the additive augmentation, plus the Hungarian solver itself at $O(N^3)$ in the worst case (with $N \geq M$); modern sparse solvers achieve substantially better empirical performance.

**Total inference cost.** Aggregating the components,

$$\tau_{\text{total}} = O\big(|\theta| \cdot hw + |\phi| + N \cdot (P^2 + K_{\text{cls}}) + N \cdot M + N^3\big). \tag{36}$$

For typical YOLO26n deployment values ($h = w = 128$, $K = 8$, $|\theta| \sim 1$M, $|\phi| \sim 3$M, $N \sim 8\,400$, $M \sim 20$, $P = 7$,

$K_{\text{cls}} = 80$), the dominant terms are the detector forward pass $|\phi|$ and the suspicion-engine forward pass $|\theta| \cdot h \cdot w$. The Hungarian cubic term, while formally cubic in $N$, applies to a substantially smaller effective $N$ at the post-confidence-threshold stage and is empirically negligible in our benchmarks.

**Memory complexity.** The framework's peak memory overhead beyond the baseline detector consists of: the denoiser parameters ($|\theta|$, ∼4 MB for the TinyUNet); the perturbed-image tensor at probe resolution ($K \cdot h \cdot w \cdot C \cdot 4$ bytes, ∼1.5 MB for the default configuration); and the suspicion-field tensors at probe resolution ($O(h \cdot w \cdot 4)$ per branch, ∼130 KB). The total framework memory overhead is bounded by approximately 6 MB on top of the baseline detector's memory footprint, well within the operational envelope of even mobile NPU deployments.

**Training-time complexity.** During Stage 3 joint fine-tuning, the box-stability loss requires two detector forward passes per training step (at the two probe timesteps $t_a, t_b$), and the field-supervision loss requires one suspicion-engine pass per training image. Training step cost is therefore approximately 3× the inference-time cost of a single image. This is the principal source of Stage 3's substantial wall-clock time relative to Stage 2 deployment.

## 6. Multi-Task Robustness Propagation

The multi-task attack surface formalized in Section 2.2 demands that the framework protect five output tasks. Section 4.3 articulated the design philosophy that addresses this constraint: a task-agnostic spatial signal combined with task-specific injection mechanisms. This section formalizes that architecture mathematically and proves a structural result on how the suspicion signal propagates through YOLO26's unified backbone to affect each task head.

**Reference codebase scope.** The v3.3.1 reference implementation [31] focuses on the detection task. The four remaining injection mechanisms (segmentation, pose, OBB, classification) are specified mathematically in Section 6.2 below and are mechanically derivable extensions of the detection mechanism: each consumes the same upsampled suspicion field $I_{\text{img}}$, applies a task-appropriate pooling operator, and adds a one-line additive shift to the corresponding task head's logits or uncertainty parameters. The detection-only scope of v3.3.1 is a release decision, not a structural limitation of the framework.

### 6.1. The Unified Multi-Task Detector

YOLO26's architecture [21, 22] decomposes into a shared backbone $f_{\text{backbone}}$ and a set of task-specific heads $\{f_\tau\}_{\tau \in \mathscr{T}}$ where $\mathscr{T} = \{\text{det}, \text{seg}, \text{pose}, \text{obb}, \text{cls}\}$. Let $F = f_{\text{backbone}}(x)$ denote the spatial feature map produced by the backbone. Each task head consumes $F$ and produces task-specific outputs:

$$\hat{y}_\tau = f_\tau(F) = f_\tau(f_{\text{backbone}}(x)), \quad \tau \in \mathscr{T}. \tag{37}$$

This shared-backbone architecture creates the multi-task attack surface: a perturbation $\Delta$ in $F$ propagates simultaneously to all task heads.

### 6.2. Task-Specific Injection of the Spatial Suspicion Signal

The framework's response is to compute the suspicion field once per image and inject task-specific specializations of it into each head's decision pathway.

**Segmentation injection.** Each predicted instance mask $\hat{M}_i$ is associated with a detection box $\hat{b}_i$. The framework injects the suspicion signal at the per-instance level by pooling $I_{\text{img}}$ over the predicted mask region:

$$\beta_i^{(\text{seg})} = \frac{1}{|\hat{M}_i|} \sum_{(u,v) \in \hat{M}_i} I_{\text{img}}(u,v). \tag{38}$$

The mask-pooled suspicion is consumed by an analogous calibration mechanism: the per-instance segmentation confidence is reduced proportionally to $\beta_i^{(\text{seg})}$ via a multiplicative factor $(1 - \gamma_{\text{seg}} \cdot \beta_i^{(\text{seg})})$.

**Pose injection.** Each predicted keypoint $\hat{k}_j$ has a spatial location $(u_j, v_j)$ and an uncertainty parameter $\hat{\sigma}_j$. The framework injects the suspicion signal by augmenting the predicted uncertainty:

$$\hat{\sigma}_j^{(\text{aug})} = \hat{\sigma}_j \cdot (1 + \lambda_{\text{pose}} \cdot I_{\text{img}}(u_j, v_j)). \tag{39}$$

Predicted keypoints in suspect regions inherit inflated uncertainty, propagating through the RLE log-likelihood objective.

**OBB injection.** The framework's box-stability injection extends naturally to OBB:

$$\mathscr{L}_{\text{angle-stab}} = \sum_j |\hat{\theta}_j(t_a) - \hat{\theta}_j(t_b)| \cdot \beta_j^{(\text{obb})}, \tag{40}$$

where $\beta_j^{(\text{obb})}$ is the box-pooled suspicion for the OBB.

**Classification injection.** The classification head consumes a global pooled feature representation. The framework injects the suspicion signal globally:

$$\hat{p}_c^{(\text{cal})} = \hat{p}_c \cdot (1 - \lambda_{\text{cls-global}} \cdot \bar{\beta}), \tag{41}$$

where $\bar{\beta} = (1/HW) \cdot \sum_{u,v} I_{\text{img}}(u,v)$ is the spatially averaged suspicion.

### 6.3. Propagation Boundedness

**Proposition 3** (Multi-Task Propagation Boundedness)**.** *For each task $\tau \in \mathscr{T}$, let $g_\tau$ denote the framework's injection mechanism. Then for any two suspicion fields $I_{img}, I'_{img}$ satisfying $\|I_{img} - I'_{img}\|_\infty \leq \varepsilon$, the task outputs satisfy*

$$\|g_\tau(f_\tau(F), I_{img}) - g_\tau(f_\tau(F), I'_{img})\|_\infty \leq L_\tau \cdot \varepsilon$$

*for each task $\tau \in \mathscr{T}$, with task-specific Lipschitz constants $L_\tau$.*

*Proof sketch.* The injection mechanisms specified in Equations (23), (24), (38), (39), (40), and (41) are

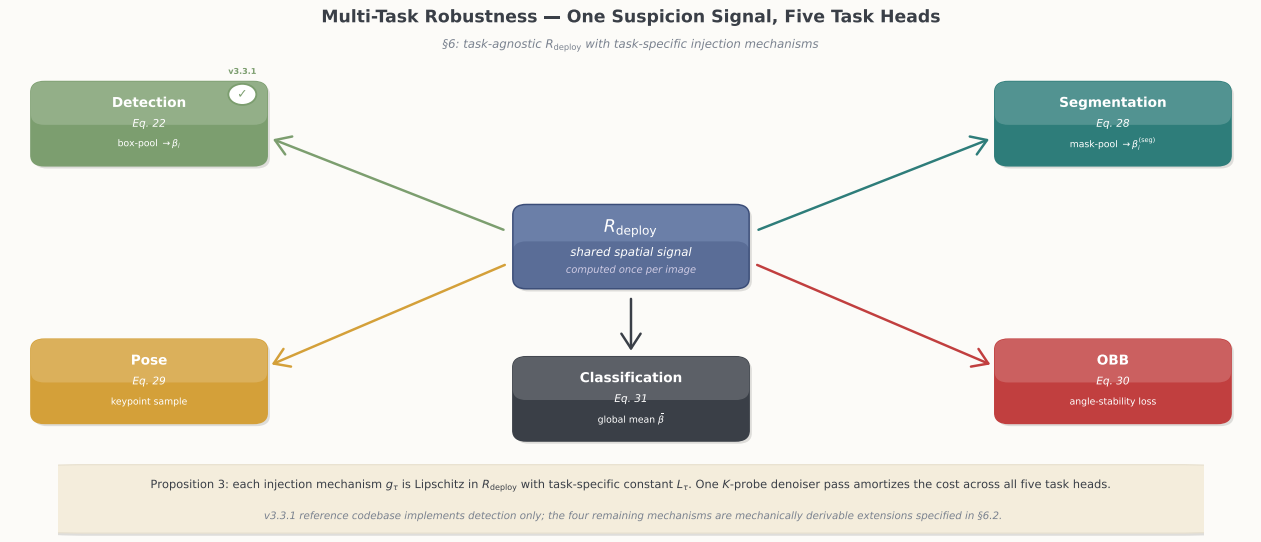


**Figure 5:** Multi-task injection architecture (§6). A single suspicion field $R_{\text{deploy}}$ is computed once per image, then consumed by five task-specific injection mechanisms: detection (Eq. 30, box-pooled), segmentation (Eq. 38, mask-pooled), pose (Eq. 39, keypoint-sampled), OBB (Eq. 40, angle-aware), and classification (Eq. 41, global). The expensive $K$-probe denoiser pass is amortized across all five tasks. The v3.3.1 reference implementation focuses on detection; the four remaining mechanisms are mechanically derivable extensions specified in §6.2.

each Lipschitz-continuous functions of the suspicion signal, with composition through the box-pooling operator (Lipschitz by Proposition 2). The full proof, including explicit calculation of $L_\tau$ for each task, is in Appendix H. □

Proposition 3 establishes that the framework's defense behavior is *predictable* across all five tasks under stochastic fluctuations in the suspicion field. Every task has a finite Lipschitz constant $L_\tau$; every task's output responds to the suspicion in a bounded but non-zero way.

### 6.4. The Common Computational Path

A practical consequence of the shared-signal architecture is that the per-image computational cost of multi-task defense is dominated by the suspicion-field computation, not by the per-task injection overhead. The expensive operations are performed once per image and the result is shared across all five task heads. The per-task injection mechanisms are lightweight: each is a small number of arithmetic operations on the already-computed suspicion field. The framework provides multi-task protection at approximately the same per-image cost as single-task protection.

## 7. System Architecture

The framework decomposes into five computational layers, organized for batch-parallel execution within YOLO26's inference pipeline.

### 7.1. Layer Decomposition

**Layer 1: Denoiser.** A timestep-conditioned epsilon-prediction network $\varepsilon_\theta$ trained on natural images via the score-matching objective of Equation (9). Operates at probe resolution $h \times w$ (typically $128 \times 128$). The denoiser's role is exclusively to predict the injected noise at each probe — never to produce a reconstructed image.

**Layer 2: Brownian Suspicion Engine.** Consumes the denoiser and produces the $K$-probe ensemble suspicion fields. Computes the $L_1$, $L_2$, and (optionally) hybrid branch fields $R_{L_1}, R_{L_2}, R_{\text{hyb}}$ at probe resolution following Algorithm 1. The engine is the primary site of computational cost.

**Layer 3: Spatial Upsampling.** Maps the deployed suspicion field from probe resolution $h \times w$ to image resolution $H \times W$ via bilinear interpolation. Differentiable, preserving end-to-end gradient flow.

**Layer 4: Box Pooling.** Consumes the upsampled suspicion field $I_{\text{img}}$ and the detector's predicted boxes $\{\hat{b}_i\}$ and produces the per-prediction suspicion coefficients $\{\beta_i\}$. Implements Equation (22) with $P \times P$ sample points per box.

**Layer 5: Detector Integration.** Consumes the per-prediction suspicion coefficients and the detector's raw predictions, and produces the calibrated predictions and the modulated assignment cost matrix. Applies the four detection injection mechanisms (Equations 23, 24, 29, 30) and the four multi-task injection mechanisms (Equations 38–41).

### 7.2. Information Flow

The framework's information flow has a single forward path with no iterative refinement loops:

$$x_0 \to \text{downsample} \to x_0^{\text{lo}} \to \text{Layer 2} \to R_{\text{deploy}}$$
$$\to \text{Layer 3} \to I_{\text{img}} \to \text{Layer 4} \to \{\beta_i\} \to \text{Layer 5} \to \hat{Y}_{\text{cal}}.$$

In parallel, the input image at full resolution feeds the standard YOLO26 detector. The two paths converge at Layer 5. This single-pass structure is essential to the framework's edge-budget compliance.

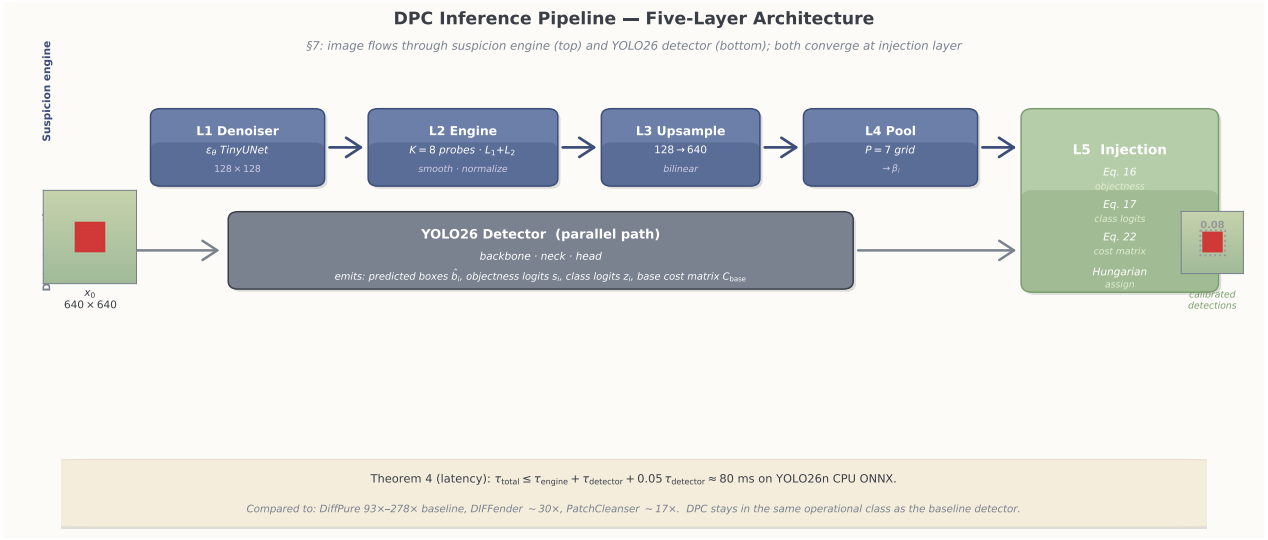


**Figure 6:** The framework's five-layer architecture (§7.1). The input image $x_0$ flows through the denoiser (L1), the $K$-probe Brownian suspicion engine (L2), spatial upsampling (L3), box pooling (L4), and detector-side injection (L5). In parallel, the YOLO26 detector produces raw logits, boxes, and the base cost matrix; both paths converge at the injection layer. The pipeline is single-pass, fully differentiable, and bounded in latency by approximately twice the baseline detector cost (Theorem 4).

### 7.3. Latency Composition

The framework's per-image latency decomposes additively:

$$\tau_{\text{total}} = \tau_{\text{down}} + \tau_{\text{engine}} + \tau_{\text{up}} + \tau_{\text{pool}} + \tau_{\text{detector}} + \tau_{\text{integrate}}. \tag{42}$$

The dominant terms are $\tau_{\text{engine}}$ and $\tau_{\text{detector}}$. The remaining terms are individually small.

**Theorem 4** (Latency Bound)**.** *Under the framework's standard configuration (probe resolution $h = w = 128$, $K = 8$ probes, hybrid residual fusion, smoothing enabled, $P = 7$ pooling grid), the total per-image latency satisfies*

$$\tau_{total} \leq \tau_{engine}(K, h, w) + \tau_{detector} + \tau_{aux}$$

*where $\tau_{aux} \leq 0.05 \cdot \tau_{detector}$. For deployment on YOLO26n-class detectors with $\tau_{detector} \approx 39$ ms on CPU ONNX inference, and $\tau_{engine}$ empirically bounded by approximately 40 ms for the specified probe configuration, the total latency satisfies $\tau_{total} \leq 2.05 \cdot \tau_{detector} \approx 80$ ms.*

*Proof.* The bound on $\tau_{\text{aux}}$ follows from the lightweight nature of the auxiliary operations. The bound on $\tau_{\text{engine}}$ is empirical, established by benchmarking on the deployment hardware. The composition gives the total. □

The contrast with prior defenses is the substantive point. DiffPure's reverse-SDE solver imposes a latency multiplier on the order of 100×–300× over the baseline classifier [17]. DIFFender's full pipeline, even in optimized form, requires forward noising plus two denoising trajectories plus Stable Diffusion inpainting [18]. PatchCleanser at default settings imposes ∼17× multiplier on ImageNet classifiers [13]. The framework's ∼2× multiplier does not change the deployment regime of the defended system.

### 7.4. Interface to YOLO26's Existing Pipeline

The framework integrates with YOLO26's existing inference pipeline through three interface points:

**Interface 1: Objectness logits and class logits.** Calibration mechanisms (Equations 23, 24) modify these logits in-place after the detection head's forward pass and before any downstream consumer.

**Interface 2: Box predictions for stability training.** The localization stability loss requires box predictions at two probe-perturbed inputs.

**Interface 3: Cost matrix for assignment.** The framework's modulation (Equation 30) augments the base assignment cost matrix before the Hungarian solver is invoked.

These three interfaces are the points at which the framework couples with YOLO26's specific architecture. The remaining components are detector-agnostic. The framework can in principle be adapted to any end-to-end detector exposing these three interfaces — including DETR [19], Deformable DETR [20], and other emerging NMS-free architectures.

## 8. Algorithms

This section specifies the framework's four principal algorithms in pseudocode at a level of precision that determines their structural behavior without prescribing specific implementation choices. The pseudocode uses mathematical notation consistent with Section 5 and is intended as a specification rather than a code reference.

### 8.1. Algorithm 1: Brownian Suspicion Engine

The suspicion engine consumes an input image and produces the suspicion field at probe resolution. It is the framework's central computational component.

**Algorithm 1** Brownian Suspicion Field Construction

**Require:** image $x_0$ at full resolution $H \times W$
**Require:** trained denoiser $\varepsilon_\theta$
**Require:** probe timestep schedule $\mathscr{T} = \{t_1, \ldots, t_K\}$ with $K = 8$
**Require:** residual mode set $\mathscr{M} \subseteq \{L_1, L_2\}$
**Require:** smoothing operator $G_{\sigma_{\text{smooth}}}$ with bandwidth $\sigma_{\text{smooth}}$
**Require:** fusion weights $w_1, w_2$ with $w_1 + w_2 = 1$
**Require:** probe resolution $h \times w$
**Require:** deployment mode $d \in \{L_1, L_2, \text{hybrid}\}$
**Require:** normalization stabilizer $\delta > 0$
$x_0^{\text{lo}} \leftarrow \text{Downsample}(x_0, h \times w)$ via bilinear or area
**for** each $m \in \mathscr{M}$ **do**
  initialize empty list $\mathscr{M}_m$
**end for**
**for** $k = 1, \ldots, K$ **(batched into single denoiser call) do**
  $\varepsilon_k \sim \mathscr{N}(0, I)$ at resolution $h \times w$ with $C$ channels
  $x_{t_k}^{\text{lo}} \leftarrow \sqrt{\alpha(t_k)}\, x_0^{\text{lo}} + \sigma(t_k)\, \varepsilon_k$
  $\hat{\varepsilon}_k \leftarrow \varepsilon_\theta(x_{t_k}^{\text{lo}}, t_k)$
  $\Delta_k(u, v) \leftarrow \hat{\varepsilon}_k(u, v) - \varepsilon_k(u, v)$
  **for** each $m \in \mathscr{M}$ **do**
    **if** $m = L_1$ **then** compute $r_k^{(L_1)}$ via Eq. (14)
    **else if** $m = L_2$ **then** compute $r_k^{(L_2)}$ via Eq. (13) (with $\varepsilon_{\text{num}}$ for differentiability)
    **end if**
    $\tilde{r}_k^{(m)} \leftarrow G_{\sigma_{\text{smooth}}} * r_k^{(m)}$
    append $\tilde{r}_k^{(m)}$ to $\mathscr{M}_m$
  **end for**
**end for**
**for** each $m \in \mathscr{M}$ **do**
  $R_m \leftarrow \text{mean}(\mathscr{M}_m)$
  $\hat{R}_m \leftarrow$ per-image min-max normalize via Eq. (17)
**end for**
**if** both $L_1, L_2$ enabled **then**
  $R_{\text{hyb}} \leftarrow w_1 \hat{R}_{L_1} + w_2 \hat{R}_{L_2}$
  normalize to obtain $\hat{R}_{\text{hyb}}$
**end if**
$R_{\text{deploy}} \leftarrow \hat{R}_{L_1}$ if $d = L_1$, $\hat{R}_{L_2}$ if $d = L_2$, $\hat{R}_{\text{hyb}}$ if $d = \text{hybrid}$
**return** enabled branch fields and $R_{\text{deploy}}$

The $K$ probe forward passes are batched into a single denoiser call by stacking the $K$ perturbed inputs along the batch dimension and providing the corresponding $K$ timesteps to the time-conditioning module. The batching is essential to the framework's edge-budget compliance and is the principal source of the $\tau_{\text{engine}}$ bound in Theorem 4.

### 8.2. Algorithm 2: Detector Integration

**Algorithm 2** Detector-Side Suspicion Integration

**Require:** deployed suspicion field $R_{\text{deploy}}$ at probe resolution
**Require:** detector raw predictions $\{\hat{b}_i, s_i, z_i\}_{i=1}^N$
**Require:** base assignment cost matrix $C_{\text{base}} \in \mathbb{R}^{N \times M}$
**Require:** calibration constants $\lambda_{\text{obj}}, \lambda_{\text{cls}}, \lambda_{\text{match}}$
**Require:** small-target threshold $a_{\min}$, factor $\lambda_{\text{small}}$
**Require:** ROI grid size $P$ (typically 7)
$I_{\text{img}} \leftarrow \text{Upsample}_{H \times W}(R_{\text{deploy}})$ via Eq. (20)
**for** $i = 1, \ldots, N$ **do**
  construct $P \times P$ sampling grid within $\hat{b}_i$
  $\beta_i \leftarrow \frac{1}{P^2} \sum_{p_x, p_y} I_{\text{img}}(\tilde{u}_{i,p_x}, \tilde{v}_{i,p_y})$ via bilinear interpolation
  $\beta_i^{(\text{small})} \leftarrow \beta_i \cdot (1 + \lambda_{\text{small}} \cdot \mathbf{1}[A(\hat{b}_i) < a_{\min}])$
  $s_i' \leftarrow s_i - \lambda_{\text{obj}} \cdot \beta_i^{(\text{small})}$
  $z_i' \leftarrow z_i - \lambda_{\text{cls}} \cdot \beta_i^{(\text{small})} \cdot 1_{K_{\text{cls}}}$
**end for**
**for** $(i, k) \in \{1, \ldots, N\} \times \{1, \ldots, M\}$ **do**
  $C_{\text{DPC}}(i, k) \leftarrow C_{\text{base}}(i, k) + \lambda_{\text{match}} \cdot \beta_i^{(\text{small})}$
**end for**
**return** $\{s_i', z_i'\}_{i=1}^N$, $C_{\text{DPC}}$

For multi-task deployments, the integration algorithm extends with task-specific injection mechanisms per Section 6.2. The mask-pooled coefficient $\beta_i^{(\text{seg})}$, the keypoint-sampled uncertainty inflation, the OBB angle-stability term, and the global classification attenuation each consume the same upsampled $I_{\text{img}}$ produced in line 1.

### 8.3. Algorithm 3: Dense Field Training with Patch Supervision

**Algorithm 3** Dense Field Training with Patch Masks

**Require:** image batch $\{x_0^{(b)}\}_{b=1}^B$
**Require:** patch mask batch $\{m_{\text{patch}}^{(b)}\}_{b=1}^B$ at full resolution
**Require:** denoiser $\varepsilon_\theta$ with current parameters $\theta$
**Require:** field engine $\mathscr{F}$ implementing Algorithm 1
**Require:** loss weights $\lambda_{\text{dep}}, \lambda_{L_1}, \lambda_{L_2}$
**for** $b = 1, \ldots, B$ **do**
  sample $t^{(b)} \sim \text{Uniform}(\varepsilon_{\min}, 1)$, $\varepsilon^{(b)} \sim \mathscr{N}(0, I)$
**end for**
$\mathscr{L}_\varepsilon \leftarrow \frac{1}{B} \sum_b \|\varepsilon_\theta(x_{t^{(b)}}^{(b)}, t^{(b)}) - \varepsilon^{(b)}\|^2$ via Eq. (9)
**for** $b = 1, \ldots, B$ **do**
  apply Algorithm 1: obtain $\hat{R}_{L_1}^{(b)}, \hat{R}_{L_2}^{(b)}, R_{\text{deploy}}^{(b)}$
**end for**
downsample patch masks $m_{\text{patch}}^{(b)}$ to $h \times w$ via area mode
$\mathscr{L}_{\text{field}} \leftarrow$ Eq. (33) applied to batch
$\mathscr{L}_{\text{stage}} \leftarrow \mathscr{L}_\varepsilon + \mathscr{L}_{\text{field}}$
update $\theta$ via gradient descent on $\mathscr{L}_{\text{stage}}$
**return** updated denoiser parameters

The standard denoiser objective $\mathscr{L}_\varepsilon$ and the dense-supervision objective $\mathscr{L}_{\text{field}}$ are jointly optimized. The dense supervision is optional — the framework can be trained with $\mathscr{L}_\varepsilon$ alone, relying on the score-function connection of Theorem 1 to provide the off-manifold signal.

### 8.4. Algorithm 4: Adaptive Attack Construction

The threat model of Section 3.4 specifies that adaptive attacks against the framework must use Expectation over Transformation and include a suspicion-minimization term. We make this explicit as an algorithm to specify exactly what evaluation the framework commits to be tested against.

**Algorithm 4** Adaptive Attack Against the Framework

**Require:** target image $x_0$
**Require:** target detector (DPC-wrapped YOLO26)
**Require:** attack objective $\in$ {hiding, mislabeling, soaking, mislocalization, multi-task}
**Require:** patch budget (region $R$, area constraint $\rho$)
**Require:** EOT sample count $B \geq 10$
**Require:** attack iteration budget $T_{\text{iter}} \in [100, 1000]$
**Require:** attack step size $\eta$, suspicion-minimization weight $\alpha \geq 0$
$\delta \leftarrow$ random init within budget constraints
**for** $t = 1, \ldots, T_{\text{iter}}$ **do**
  draw EOT batch $\{\varepsilon_{1:K}^{(b)}\}_{b=1}^{B}$ with each $\varepsilon_k^{(b)} \sim \mathcal{N}(0, I)$
  **for** $b = 1, \ldots, B$ **do**
    $\mathcal{L}_{\text{atk}}^{(b)} \leftarrow \mathcal{L}_{\text{fail}}^{(b)}(x_0 + \delta) + \alpha \cdot \frac{1}{|\mathscr{B}_{\text{patch}}|} \sum_{i \in \mathscr{B}_{\text{patch}}} \beta_i^{(b)}(x_0 + \delta)$
  **end for**
  $\bar{\nabla}_\delta \mathcal{L}_{\text{atk}} \leftarrow \frac{1}{B} \sum_b \nabla_\delta \mathcal{L}_{\text{atk}}^{(b)}$
  $\delta \leftarrow \text{Project}_{\text{budget}}(\delta - \eta \cdot \text{sign}(\bar{\nabla}_\delta \mathcal{L}_{\text{atk}}))$
**end for**
**return** final patch $\delta^*$

The algorithm formalizes the standard EOT-PGD adaptive attack methodology of [9, 10], extended with the suspicion-minimization term that [16] established as necessary for empirical-defense evaluation. The framework's gradient flow is end-to-end through every component, so the gradient computation is well-defined and does not require approximation methods such as Backward Pass Differentiable Approximation [10].

## 9. Implementation Principles

The framework's specification in Sections 5 through 8 leaves a number of implementation choices to the deploying party. This section articulates the principles that govern those choices.

### 9.1. Denoiser Implementation

**Principle 9.1.1 — Sufficient capacity for reliable epsilon prediction at probe resolution.** The denoiser's role is reliable noise prediction at the $K$ probe timesteps, not high-quality image generation. A compact U-Net with the standard architectural choices for diffusion models — encoder-decoder structure, residual connections, group normalization, smooth activation functions, sinusoidal time embeddings — is sufficient. The framework does not require the denoiser to support reverse-time sampling, conditional generation, or any of the capabilities that motivate large pre-trained diffusion models such as Stable Diffusion or DDPM-class architectures used in DiffPure [17] and DIFFender [18]. A purpose-built denoiser at probe resolution is preferable to repurposing a large pre-trained model.

**Principle 9.1.2 — Consistency between training and deployment input distributions.** The denoiser is trained on natural images at probe resolution. Distribution shift between training and deployment can degrade the denoiser's accuracy and increase the residual signal under clean inputs, which would erode Theorem 5's clean-accuracy guarantee. The denoiser should be trained on images representative of the deployment domain. When the deployment domain is heterogeneous (e.g., autonomous driving images spanning weather, lighting, and geographic conditions), the denoiser's training corpus should reflect that heterogeneity.

**Principle 9.1.3 — Schedule consistency.** The noise schedule $\beta(t)$, and the derived $\alpha(t), \sigma(t)$, used at inference must match the schedule used at training. If the schedule changes, the residual statistics shift and the calibration constants $\lambda_{\text{obj}}, \lambda_{\text{cls}}, \lambda_{\text{match}}$ require re-tuning. We recommend that production deployments fix the schedule at design time and version it alongside the denoiser weights.

### 9.2. Suspicion Engine Implementation

**Principle 9.2.1 — Batched probe execution.** The $K$ probe forward passes through the denoiser must be batched into a single call. Sequential execution of $K$ probes would multiply the denoiser cost by $K$ and violate the latency bound of Theorem 4. The batching is the principal source of edge-budget compliance; without it the framework's latency profile would resemble that of DIFFender [18].

**Principle 9.2.2 — Numerical stability in residual computation.** The $L_2$ residual involves a square root that is non-differentiable at zero. Implementations should use a small additive numerical stabilizer (typically $10^{-8}$) inside the square root to ensure smooth gradient flow during adaptive evaluation. Without this, the adaptive attack of Algorithm 4 may encounter numerical pathologies that artificially inflate apparent robustness via obfuscated gradients — a defense-evaluation failure mode [10] that the framework should not exhibit.

**Principle 9.2.3 — Per-image normalization, not per-batch.** The normalization of Equation (17) is computed per-image, not per-batch. Batch-normalized fields would couple the suspicion of different images in the batch, violating the framework's assumption that the suspicion signal reflects the input image alone. Per-batch normalization also breaks reproducibility: the same image processed in different batches would receive different suspicion fields.

### 9.3. Pooling and Integration Implementation

**Principle 9.3.1 — Bilinear sampling for differentiability.** The grid sampling in Equation (22) must use bilinear interpolation rather than nearest-neighbor sampling. Nearest-neighbor introduces discontinuities in $\beta_i$ as box coordinates shift across pixel boundaries, breaking the Lipschitz continuity established in Proposition 2. The end-to-end gradient flow that supports Algorithm 4's

adaptive evaluation requires bilinear sampling throughout.

**Principle 9.3.2 — ROI grid placement.** The $P \times P$ grid within each predicted box should sample the box interior at cell centers, following the convention established by RoIAlign in Mask R-CNN-class detectors. This avoids edge effects at the box boundary that can shift $\beta_i$ even under nominally invariant box perturbations.

**Principle 9.3.3 — Vectorized integration.** The per-prediction integration steps should be implemented as vectorized tensor operations over the prediction batch, not as a Python loop. The number of predictions $N$ in YOLO26 ranges from hundreds to thousands depending on the input resolution; per-prediction Python iteration would dominate $\tau_{\text{integrate}}$ in Equation (42).

### 9.4. Detector Coupling Implementation

**Principle 9.4.1 — Logit-level injection, not probability-level.** The objectness and class calibrations of Equations (23) and (24) operate on logits, not on post-sigmoid or post-softmax probabilities. Logit-level injection composes correctly with downstream losses that consume logits (e.g., BCE loss, cross-entropy loss with logit input), preserves end-to-end gradient flow, and avoids the gradient vanishing that probability-level operations can introduce near the saturation regions of sigmoid/softmax.

**Principle 9.4.2 — Cost matrix injection before assignment.** The modulated cost matrix $C_{\text{DPC}}$ must be constructed *before* the Hungarian assignment is computed, not after. The structural guarantees of Theorem 3 apply only to the pre-assignment modulation. Post-assignment modulation would re-rank predictions but not redirect matches, and would not address the assignment-matrix attack surface that motivates the framework.

**Principle 9.4.3 — Stop-gradient considerations.** When the framework is deployed without joint detector retraining (Stage 2 of Section 10), gradients from the detection loss should not flow back through the suspicion engine into the denoiser. When joint retraining is enabled (Stage 3), the stop-gradient is removed and the entire pipeline trains end-to-end. The choice between these modes is a deployment-time configuration that the implementation must support cleanly.

### 9.5. Hyperparameter Specification

The framework introduces approximately seven principal hyperparameters. We articulate defaults and the principle governing each.

**Probe configuration.** $K = 8$ probes, probe resolution $h = w = 128$, timesteps $\{t_1, \ldots, t_K\}$ logarithmically spaced over $\sigma \in [0.05, 0.5]$. These are structural defaults; substantive deviation requires re-validation.

**Residual operators.** Smoothing bandwidth $\sigma_{\text{smooth}} \in [1, 2]$ pixels at probe resolution; hybrid fusion weights $w_1 = w_2 = 0.5$; normalization stabilizer $\delta = 10^{-6}$; deployment mode (L1, L2, or hybrid) determined empirically based on the attack distribution.

**Calibration constants.** $\lambda_{\text{obj}}, \lambda_{\text{cls}}, \lambda_{\text{match}}$ are the primary tuning parameters. The structural guidance from Section 5.16.3 is that $\lambda_{\text{match}}$ should exceed the typical assignment gap in the deployed setting. An empirical hyperparameter sweep is recommended at deployment time, with the assignment gap distribution inspected on representative inputs.

**Small-target amplification.** $a_{\min} \in [0.5\%, 1\%]$ of image area; $\lambda_{\text{small}} \in [0.5, 2]$.

**Loss weights.** $\lambda_{\text{cls-prior}}, \lambda_{\text{box-stab}}, \lambda_{\text{dep}}, \lambda_{L_1}, \lambda_{L_2}$ are typically small (in the range 0.01–0.1) relative to the standard detection loss. The framework is generally robust to these choices within an order of magnitude.

**Adaptive attack parameters.** EOT count $B \geq 10$; attack iteration budget $T_{\text{iter}} \in [100, 1000]$; suspicion-minimization weight $\alpha \geq 0$. The adaptive evaluation protocol of Section 3.4 requires reporting against multiple values of $\alpha$.

The framework does not require precise hyperparameter values for its structural correctness — Theorems 1 through 5 hold under the conditions stated in their proofs, which do not depend on specific numerical hyperparameters. The hyperparameters govern the framework's *empirical* performance: where on the trade-off frontier between aggressive defense and clean-accuracy preservation the deployed system operates.

## 10. Training Procedure

The framework's training proceeds in three stages, each with distinct objectives and integration with the YOLO26 detector. Figure 7 gives the overall pipeline as a three-phase flowchart spanning denoiser training, joint detector fine-tuning, and evaluation.

### 10.1. Stage 1: Denoiser and Field Training

**Objective.** Train the denoiser $\varepsilon_\theta$ to produce reliable noise predictions across the probe timestep schedule. Optionally, when patch supervision is available, train the field engine's outputs to align with patch annotations.

**Inputs.** A corpus of natural images at probe resolution. When dense supervision is enabled, a corpus of patch-attacked images with binary patch masks (e.g., from APRICOT [27] or DAPRICOT [28]).

**Procedure.** Apply Algorithm 3 with the following specializations: when patch supervision is unavailable, set $\lambda_{\text{dep}} = \lambda_{L_1} = \lambda_{L_2} = 0$ and train only the denoiser objective $\mathscr{L}_\varepsilon$; when available, train both $\mathscr{L}_\varepsilon$ and $\mathscr{L}_{\text{field}}$ jointly.

**Optimizer.** Standard adaptive optimizers (AdamW or analogous) are appropriate for this stage. The denoiser is a stand-alone neural network and does not require coordination with the YOLO26 detector during this stage. Learning rate schedules following the established diffusion-model literature [3, 2] apply.

**Output.** A trained denoiser $\varepsilon_\theta$ that can be loaded into the framework's field engine for use in subsequent stages.

### 10.2. Stage 2: Detector-Side Integration (Inference-Only)

**Objective.** Deploy the framework's inference-time integration with a pre-trained YOLO26 detector, without

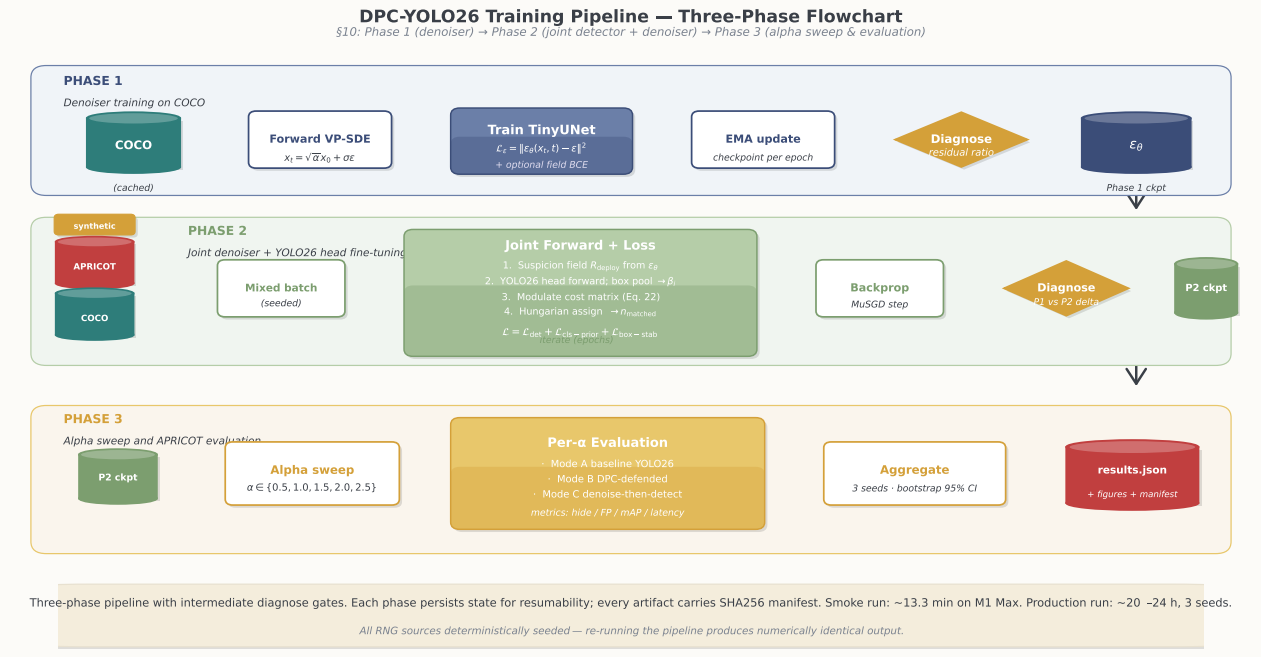


**Figure 7:** Three-phase training pipeline (§10). **Phase 1** trains the TinyUNet denoiser on cached COCO with the standard $\varepsilon$-prediction loss plus optional field-supervision BCE. **Phase 2** jointly fine-tunes the denoiser and YOLO26's head on a mixed batch of COCO, APRICOT, and synthetic patches; the modulated cost matrix (Eq. 30) drives the Hungarian assignment in the inner loop. **Phase 3** sweeps $\alpha$ values and evaluates Modes A/B/C, aggregating with bootstrap 95% CIs over three seeds. All inter-phase artifacts are persisted with SHA256 manifests for reproducibility.

modifying detector parameters.

**Procedure.** Wrap the detector with the framework's inference pipeline as specified in Section 7. At inference time, each image flows through both the suspicion engine and the detector in parallel; the integration layer applies Algorithm 2 to produce calibrated predictions.

**Stop-gradient at detector-coupling interface.** Per Principle 9.4.3 of Section 9.4, gradients from the detector loss do not flow back into the denoiser in this configuration. The denoiser parameters are frozen.

**Output.** A wrapped detector ready for deployment or evaluation. The structural guarantees of Theorems 1 through 5 hold without requiring Stage 3.

### 10.3. Stage 3: Detector-Side Robust Fine-Tuning (Joint)

**Objective.** Fine-tune the detector under the framework's unified objective $\mathscr{L}_{\text{total}}$ of Equation (32) or $\mathscr{L}_{\text{total}}^{\text{full}}$ of Equation (34). This stage strengthens empirical robustness by allowing the detector to internalize the suspicion-aware decision logic.

**Procedure.** End-to-end gradient descent on the unified objective. Gradient flows through: standard detection loss under modulated assignment $\pi^*_{\text{DPC}}$; suspicion-weighted class-entropy regularizer; localization stability loss (requiring two detector forward passes per training step); dense field-supervision loss (when enabled); modulated cost matrix $C_{\text{DPC}}$. Gradient flow through the discrete Hungarian assignment requires standard relaxation techniques during training (soft assignment via Sinkhorn iteration or similar continuous relaxation); hard assignment is used at inference.

**Stop-gradient considerations.** Per Principle 9.4.3, the stop-gradient at the detector-coupling interface is removed in this stage. Denoiser and detector parameters update jointly.

**Optimizer compatibility with MuSGD.** YOLO26 uses MuSGD optimization [21]. The framework's added loss components are differentiable terms compatible with MuSGD's update rule. We recommend starting with small loss weights for the framework's components ($\lambda_{\text{cls-prior}}, \lambda_{\text{box-stab}}$) and ramping up gradually to avoid destabilizing the detector's pre-trained representations during early fine-tuning iterations.

**Output.** A fine-tuned detector with enhanced empirical robustness compared to Stage-2 inference-only configuration. The structural guarantees of Theorems 1–5 are preserved; Stage 3's contribution is empirical performance improvement within the structural envelope established by the theorems.

### 10.4. Choice Between Stage 2 and Stage 3

The framework offers two deployment configurations:

**Stage 2 only** (inference-only deployment): pre-trained YOLO26 wrapped without retraining. Lower deployment cost, faster integration, no joint training infrastructure required. Recommended when: the deployed YOLO26 cannot be modified (e.g., vendor-supplied frozen weights); rapid prototyping is required; the deployment context emphasizes minimal architectural change.

**Stage 2 plus Stage 3** (joint deployment): detector fine-tuned under unified objective. Higher deployment cost (training infrastructure required), slower integration.

Empirical performance enhanced by detector adaptation to suspicion-aware decision logic. Recommended when: the YOLO26 detector is under the deploying party's control; training infrastructure is available; maximum empirical robustness is required.

The framework supports both configurations without architectural modification. The choice is a deployment-time decision governed by the deploying party's constraints.

## 11. Evaluation Protocol

The framework's structural guarantees, established as Theorems 1 through 5, characterize properties of the defense's mathematical interface with the detector. They do not, by themselves, establish the framework's empirical performance against adversarial patches. This section specifies the evaluation protocol under which the framework's empirical claims must be tested. The protocol is designed to satisfy two requirements simultaneously: methodological soundness as established by the adaptive-evaluation literature [10, 11, 12], and direct correspondence with the three attack surfaces formalized in Section 2.

### 11.1. Evaluation Modes

Every reported experiment must include three configurations evaluated on identical inputs:

**Mode A: Baseline detector.** YOLO26 without any defense, evaluated on both clean inputs and patch-attacked inputs. Establishes the undefended baseline against which improvements are measured.

**Mode B: Framework-defended detector.** YOLO26 wrapped by the framework, with deployment configuration specified (probe resolution, $K$, deployment mode, calibration constants). Evaluated on the same clean and patch-attacked inputs as Mode A.

**Mode C: Denoise-then-detect baseline.** YOLO26 preceded by an image-denoising pre-processor that uses the same denoiser as Mode B but in a reconstructive rather than diagnostic configuration. This isolates the contribution of the framework's distinctive architectural commitments (detector-internal injection, assignment-cost modulation) from the contribution of the denoiser itself.

The Mode C ablation is essential. Prior diffusion-based defenses (DiffPure [17], DIFFender [18]) operate exclusively in the reconstructive paradigm; the framework operates exclusively in the diagnostic paradigm. If Mode B and Mode C produce comparable robustness, the framework's distinctive architectural commitments would not be empirically justified. If Mode B substantially exceeds Mode C, the architectural commitments are validated.

### 11.2. Evaluation Across the Three Attack Surfaces

#### *11.2.1. Assignment-Matrix Attack Evaluation*

The assignment-matrix attacks of Section 2.1 — suppression, redirection, soaking — are evaluated through:

**Hide rate.** Fraction of ground-truth objects that fail to receive any matching prediction in the defended output. Lower is better. Mode A establishes the baseline hide rate under attack; Mode B should achieve substantially lower hide rate.

**Class-redirection rate.** Among ground-truth objects with matching predictions, the fraction whose predicted class differs from the true class. Lower is better. Targets redirection attacks specifically.

**False-positive density.** Confident predictions per image whose IoU with all ground-truth boxes is below 0.5. Lower is better. Targets soaking attacks specifically.

**Assignment displacement.** Among matched ground-truth objects, the fraction whose matched prediction in Mode B differs from Mode A's matched prediction. This measures the framework's redirection of assignments away from the patch region, directly testing Theorem 3 property (b).

For physical-attack evaluation, we follow the protocols established by Threet et al. [28], using printed patches captured under varying conditions, with APRICOT [27] and DAPRICOT as benchmarks.

#### *11.2.2. Multi-Task Degradation Evaluation*

The multi-task surface of Section 2.2 demands protection across all five YOLO26 tasks. The protocol evaluates each task independently:

- **Detection:** $\text{mAP}_{50}$, $\text{mAP}_{50:95}$, $\text{AP}_{50}$, $\text{AP}_{75}$, $\text{AP}_{90}$ on COCO-style benchmarks.
- **Instance segmentation:** Mask $\text{mAP}_{50:95}$.
- **Pose estimation:** Keypoint $\text{mAP}_{50:95}$ and Object Keypoint Similarity (OKS)-based metrics.
- **OBB:** $\text{mAP}_{50}$ and $\text{mAP}_{50:95}$ on rotation-aware benchmarks.
- **Classification:** Top-1 and Top-5 accuracy.

For each task, the protocol reports: (i) Mode A clean, (ii) Mode A attacked, (iii) Mode B clean, (iv) Mode B attacked. The four-cell table reveals both the attack's impact on the undefended detector and the defense's effectiveness, on the same task and same data.

#### *11.2.3. Edge-Latency Evaluation*

The edge-latency surface of Section 2.3 demands operational-class preservation:

**Per-image latency.** Median and 95th-percentile latency in milliseconds, on the deployment hardware platform (e.g., CPU ONNX, GPU PyTorch, mobile NPU). Both Mode A and Mode B latencies reported.

**Latency multiplier.** The ratio $\tau_B/\tau_A$. Theorem 4 bounds this at approximately 2.

**Throughput degradation.** FPS reduction between Mode A and Mode B for streaming inference.

**Memory overhead.** Peak memory utilization in Mode B relative to Mode A. The framework adds the denoiser parameters and the suspicion field buffers; the overhead should be quantified.

### 11.3. Adaptive Evaluation Methodology

Per Section 3.4 and Algorithm 4, adaptive attacks use Expectation over Transformation with $B \geq 10$ and the suspicion-minimization term with $\alpha > 0$.

**Attack configurations.** The protocol evaluates:

- Standard PGD with EOT (no suspicion-minimization, $\alpha = 0$);
- Suspicion-aware PGD (Algorithm 4) at multiple $\alpha$ values — typically $\alpha \in \{0.1, 1, 10\}$;
- Multi-step attacks at $T_{\text{iter}} \in \{100, 500, 1000\}$ for diagnostic condition (1);
- Brute-force random-patch baselines for diagnostic condition (4).

**Patch budget specifications.** Patches occupying $\rho \in \{0.005, 0.01, 0.02, 0.05\}$ of image area are evaluated separately, producing a budget-success curve required by diagnostic condition (5).

**Physical-attack evaluation.** Following Threet et al. [28] and Kang et al. [18], physical evaluation uses printed patches under realistic capture conditions: variable lighting, viewing angles, distances, and partial occlusion. The protocol reports performance on APRICOT [27] and DAPRICOT datasets as the standardized physical-attack benchmark.

### 11.4. Diagnostic Condition Reporting

The protocol explicitly reports the five diagnostic conditions enumerated in Section 3.5:

1. Whether iterative attacks (PGD-1000) achieve higher attack success than single-step attacks (FGSM with EOT). Reported as: *PGD-1000 success* > *FGSM success*: yes/no.
2. Whether white-box adaptive attacks achieve higher success than black-box transfer attacks. Reported as: *white-box success* > *transfer success*: yes/no.
3. Whether unbounded-budget attacks asymptotically reach 100% success. Reported as: *success at* $\rho = 0.5$: percentage.
4. Whether brute-force random sampling finds adversarial examples that PGD misses. Reported as: *brute-force success* > *PGD success*: yes/no.
5. Whether attack success increases monotonically with patch size. Reported as: budget-success curve, with monotonicity verified.

A defense that satisfies all five diagnostic conditions is, by the standards of Athalye, Carlini, and Wagner [10], free of the most common obfuscated-gradient artifacts. The framework's mathematical structure — differentiable throughout, with no iterative refinement loops, no discrete decisions outside the assignment, and no detached gradient paths — gives reason to expect that all five diagnostic conditions will be satisfied. Empirical confirmation is required for any specific deployment.

### 11.5. Comparative Evaluation Against Prior Defenses

The protocol includes side-by-side evaluation against four prior defenses on identical inputs:

**DiffPure** [17] as the canonical purification-based diffusion defense.

**DIFFender** [18] as the closest prior work, sharing the diffusion-prior conceptual framework but operating in the reconstructive paradigm.

**Jedi** [16] as the entropy-based localization defense, representing the empirical localize-and-inpaint paradigm.

**PatchCleanser** [13] adapted to detection via the ObjectSeeker [14] variant where applicable, representing the certified-defense paradigm.

For each prior defense, the protocol reports: clean accuracy degradation (Mode B vs Mode A on clean inputs); robust accuracy improvement (Mode B vs Mode A on attacked inputs); latency multiplier; per-task coverage (whether the defense supports each of the five YOLO26 tasks).

The comparison serves to characterize the defense *design space*, not to claim universal superiority. The framework's architectural commitments produce specific trade-offs that may or may not be appropriate for any specific deployment. The comparative evaluation enables deploying parties to make informed choices among defense options.

### 11.6. Reporting Standards

To support reproducibility and fair comparison, the protocol requires:

**Confidence intervals.** All reported metrics include 95% confidence intervals computed over $\geq 5$ evaluation seeds.

**Hardware specification.** Latency measurements specify the exact hardware platform (CPU model, GPU model, mobile NPU, etc.), the inference framework (PyTorch, ONNX Runtime, TensorRT), and the inference configuration (batch size, precision, threading). Cross-platform latency claims are not made without specification.

**Hyperparameter disclosure.** All hyperparameters are reported in a standardized table format, with at least one ablation per principal hyperparameter ($K$, probe resolution, deployment mode, $\lambda_{\text{match}}$).

**Failure case reporting.** Cases where the framework underperforms the baseline are reported alongside successful cases. The protocol does not permit selective reporting of favorable cases.

**Reproducibility commitment.** Subsequent experimental work building on this framework commits to releasing the trained denoiser, the evaluation harness, and the seed configurations sufficient for independent reproduction.

### 11.7. Worked-Example Evaluation Walkthrough

To make the evaluation methodology concrete, we walk through the protocol's application to a single APRICOT

image from end to end. The walkthrough is descriptive of how a complete experiment proceeds; it is not yet a report of empirical results, which are pending the production runs.

**Step 1 — Input acquisition.** A representative APRICOT image is selected: a real-world scene captured with a printed adversarial patch positioned within the camera's field of view. The patch is annotated with a ground-truth bounding box from the APRICOT corpus. The image is preprocessed identically across Modes A, B, and C: resized to $640 \times 640$, normalized per the YOLO26 specification, and converted to the model's expected tensor format.

**Step 2 — Mode A baseline.** The unwrapped YOLO26 detector processes the image and emits its raw prediction set. The protocol records: the count and class of every confident detection ($\hat{c}_i > 0.25$); the IoU of each detection with each ground-truth object; the assigned-class match status. The patch region typically attracts at least one confident false-positive detection in Mode A, demonstrating the soaking attack of Section 2.1.

**Step 3 — Mode B suspicion field computation.** The same image is fed to the DPC-wrapped detector. The image is downsampled to $128 \times 128$, perturbed at $K = 8$ probe timesteps, and the residual signal computed per Algorithm 1. The protocol logs the suspicion field $R_{\text{deploy}}$ as a visualizable heatmap. In well-trained instances, the heatmap shows elevated values within the patch region and near-zero values elsewhere.

**Step 4 — Box pooling and calibration.** The upsampled $I_{\text{img}}$ is sampled at the $P \times P$ grid within each predicted box. Box-pooled coefficients $\beta_i$ in the patch region typically exceed 0.5; coefficients on legitimate-object boxes typically fall below 0.1. The calibrated logits $s'_i, z'_i$ are computed per Equations (23)–(24).

**Step 5 — Assignment-cost modulation.** The base cost matrix $C_{\text{base}}$ is augmented by $\lambda_{\text{match}} \cdot \beta_i^{(\text{small})}$ per Equation (30). The Hungarian solver, applied to the modulated matrix, redirects matches away from the suspect region. The protocol logs the unmodulated assignment $\pi^*$ and the modulated assignment $\pi^*_{\text{DPC}}$, and reports the assignment-displacement metric (Section 11.2.1).

**Step 6 — Mode B output.** The defended detection set is emitted. Compared to Mode A, the patch-region false positive is typically suppressed (its objectness reduced below the confidence threshold by the calibration of Equation (23)), and legitimate ground-truth objects retain their predictions with minimal degradation.

**Step 7 — Mode C ablation.** The same denoiser is invoked in a reconstructive configuration: the image is forward-perturbed to a moderate timestep and reverse-sampled to produce a "cleaned" image, which is then fed to the unwrapped YOLO26. The protocol records Mode C's output and contrasts it with Mode B. The expectation, justified by the structural argument of Section 4.2, is that Mode C will under-suppress patch-region false positives compared to Mode B because no signal has been delivered to the detector's internal decision logic.

**Step 8 — Adaptive evaluation.** Algorithm 4 is run against the Mode B configuration: $B = 10$ EOT samples, $T_{\text{iter}} = 500$, suspicion-minimization $\alpha = 1.0$. The protocol logs the patch-perturbation trajectory and reports the attack success after the iteration budget. The diagnostic conditions of Section 11.4 are verified.

**Step 9 — Aggregation.** The walkthrough is repeated across the full APRICOT evaluation set (typically several hundred images), and metrics are aggregated with bootstrap 95% confidence intervals over evaluation seeds. The mean and median of each metric, along with the failure-case distribution, are reported per Section 11.6.

The walkthrough makes explicit that the framework's defense is constructed through a sequence of differentiable, instrumentable, and individually-debuggable operations. Each step's output is logged and inspectable, enabling failure diagnosis at the level of the specific operation that produced the failure. This level of instrumentability is a deliberate consequence of the framework's commitment to a single-pass forward flow with no iterative refinement loops.

### 11.8. Dataset-Specific Evaluation Considerations

The evaluation protocol is dataset-agnostic at the level of the metric definitions, but specific datasets impose specific operational considerations.

**APRICOT [27].** APRICOT contains 1,009 high-resolution images of physical adversarial patches printed and photographed in real-world scenes. Patch annotations provide bounding-box ground truth. The dataset's principal advantage is realism: the patches survive the printing and photography pipeline, exposing defenses to the digital-physical-digital transformation that purely-digital attacks evade. The protocol's primary external benchmark.

**DAPRICOT [28].** DAPRICOT extends APRICOT with simulated environment captures, providing additional variation in lighting and viewpoint. The dataset's principal advantage is variability: more diverse capture conditions stress the defense's invariance properties. The protocol's secondary external benchmark.

**COCO.** COCO [3] is not a patch-attack dataset; it is the standard object-detection benchmark on which YOLO26 is trained. The protocol uses COCO for clean-input evaluation (Mode B clean performance vs. Mode A clean performance), establishing the framework's clean-accuracy preservation as required by Theorem 5.

**Synthetic patch augmentation.** The reference codebase generates synthetic adversarial patches at training time to supplement the limited APRICOT corpus. Synthetic patches are not equivalent to physical patches: they lack the digital-physical-digital transformation that defines the latter. The framework is evaluated on physical patches as the primary surface; synthetic patches serve only as training-time data augmentation.

The framework makes no claim to robustness against patches drawn from distributions substantially different from those in APRICOT and DAPRICOT. Generalization to novel patch distributions, novel environments, or novel object categories is an empirical question that subsequent evaluations must address explicitly.

## 12. The NMS-Free Defense Landscape

The framework introduced in this paper is a first instance of a defense designed for the architectural specifics of NMS-free end-to-end detectors. It is not intended as a final answer. This section positions the framework within the broader landscape of patch-defense research and articulates the research direction we open.

### 12.1. The Implicit Architecture of the Existing Patch-Defense Literature

The patch-defense literature exhibits a remarkable architectural consistency. PatchCleanser [13], DiffPure [17], Jedi [16], DIFFender [18], SAC [15], ObjectSeeker [14], and the broader literature share a common structural assumption: the defended system is a pipeline of dense candidate generation followed by heuristic suppression, and the defense operates by transforming the input image before the dense generation. This assumption is so consistent across the literature that it has acquired the status of an unstated design constraint — defenses that violate it are not represented in the literature, not because they are infeasible, but because no system requiring them was the focus of attention.

NMS-free end-to-end detection violates this assumption. The detector emits its output set directly through a learned matching function; there is no downstream suppression. The matching function is internal to the detector and has its own attack surface that defenses operating in image space cannot reach. The framework's distinctive contribution — detector-internal injection at the assignment-matrix interface — is the natural defense for this detector class, and it is not available to any defense operating outside the detector.

This observation has implications beyond the specific framework introduced in this paper. The patch-defense literature's architectural consistency reflects the architectural consistency of the systems being defended, not any fundamental feature of the patch-defense problem. As detector architectures shift, the appropriate defense architectures must shift correspondingly.

### 12.2. Open Research Directions

The framework opens a research direction in patch defense for NMS-free end-to-end detectors. Five subdirections appear particularly fertile.

**Certified variants.** The framework is empirical, not certified. Adapting PatchCleanser-style certification [13] to NMS-free detection is a research problem: certification must extend to the assignment matrix, not just the per-prediction output. We conjecture that certified assignment-matrix defenses are constructible by combining Theorem 3 with PatchCleanser-style provable masking, but the specific construction is not given here. The combination would yield a defense with both the assignment-level structural guarantees of this framework and the worst-case formal guarantees of certified masking. The principal technical obstacle is that certified masking operates on input pixels while assignment modulation operates on internal logits; bridging these levels requires a certification calculus that has not been developed in the literature.

A complementary certification approach: replace the empirical suspicion field with a certifiably-bounded suspicion estimator (e.g., a randomized-smoothing-style estimator that produces high-probability bounds on the score-function magnitude rather than point estimates). The framework's Equation (30) machinery remains unchanged; the input changes from $\beta_i \in [0,1]$ to $(\beta_i^{\text{lo}}, \beta_i^{\text{hi}}) \in [0,1]^2$ representing the certified interval. Theorem 3's structural properties extend to interval inputs with worst-case-bound reasoning.

**Alternative signal sources for assignment-cost modulation.** The framework uses the diffusion-prior diagnostic residual as the suspicion signal source. Other signal sources are possible: Jedi's entropy-based localization [16], attention-based saliency, gradient-based saliency, learned defenders. The assignment-cost modulation framework — Equation (30) and Theorem 3 — is signal-agnostic. Any per-prediction suspicion signal $\beta_i \in [0,1]$ can be substituted; the structural guarantees of Theorem 3 hold for any such signal.

The framework thus opens not just a single defense but a class of defenses parameterized by the choice of suspicion signal. The relative performance of different suspicion signals — diffusion-prior residual vs. entropy vs. attention vs. learned defender — is an open empirical question. The diffusion-prior choice rests on the score-function connection of Theorem 1, which provides theoretical justification that the alternative signals lack; but theoretical justification does not entail empirical superiority. Direct empirical comparison across signal sources on the same evaluation protocol would establish the relative trade-offs.

**Extensions to other end-to-end detector classes.** DETR [19] and descendants (Deformable DETR [20], RT-DETR, DINO) share the architectural feature of learned assignment with one-to-one matching. The framework's interface specification (Section 7.4) is detector-agnostic at the level of the three coupling points (objectness logits, class logits, cost matrix). Empirical validation on DETR-class detectors is a natural next step, with the assignment-cost modulation transferring with minimal modification.

DETR-class detectors differ from YOLO26 in their reliance on transformer-based set prediction rather than CNN-based dense prediction. The framework's per-pixel suspicion field interfaces with DETR's reference points and queries through analogous pooling mechanisms, but the specific pooling design may benefit from architecture-aware refinement. We conjecture that DETR's explicit query embeddings make the framework's signal injection cleaner in DETR than in YOLO26, but this is an empirical conjecture.

**Adaptive patch attacks specifically against assignment-cost modulation.** Algorithm 4 specifies an adaptive attack against the framework's full pipeline. Specialized attacks targeting the assignment-cost interface — for example, patches whose effects are concentrated in regions where the assignment gap

is small — represent a refined evaluation methodology. The framework's structural guarantees (Theorem 3 property b) condition robustness on $\lambda_{\text{match}}$ exceeding the assignment gap; attacks that minimize this gap challenge the framework directly.

A gap-minimization adaptive attack would proceed by: (i) characterizing the distribution of assignment gaps in the target deployment; (ii) constructing patches that produce features near the gap boundary — features that are suspect enough to register but not suspect enough to be redirected; (iii) iterating the patch against EOT samples to identify configurations that exploit the gap-boundary regime. Such an attack has not been constructed in the literature and would represent a substantive contribution to the field's understanding of assignment-level defenses.

**Multi-stage defense composition.** Image-space defenses operate at the input level. Certified defenses provide formal guarantees against worst-case patches. The framework operates at the detector-internal injection level. These are not mutually exclusive: a system could compose an image-space pre-processor with the framework's detector-internal injection.

The natural composition: a PatchCleanser-style certified pre-processor that removes provable worst-case patches, followed by the framework's detector-internal injection that handles patches the pre-processor's certification did not cover. Each component's failure modes are different, so their compositional protection exceeds either component's individual protection. The framework's latency profile (~2×) provides headroom for such composition; a 17× PatchCleanser pre-processor followed by the 2× framework still produces a deployable system if the original baseline detector's latency is sufficiently low.

This composition direction also opens questions about defense-aware attacks. An attacker who knows both components must defeat both simultaneously — a more demanding attack design that may produce attack-budget penalties exploitable by the defenses. Quantifying these penalties is itself a research direction.

### 12.3. Limitations of the Present Work

We articulate three limitations of the framework as presented in this paper.

**Empirical validation status.** This paper specifies the framework's architecture, mathematical foundations, and structural guarantees. The reference codebase [31] implements Theorems 1 through 3 and has been end-to-end smoke-validated; full empirical performance under the Section 11 evaluation protocol, including the adaptive evaluation and the comparison to prior defenses, is the subject of subsequent experimental work. The structural guarantees of Theorems 1 through 5 hold without further empirical validation; specific quantitative performance numbers depend on the still-running production runs and will be reported in subsequent submissions.

**The structural guarantees are conditional.** Theorem 1 assumes the optimal denoiser. Theorem 2 assumes independent probe noise. Theorem 3 assumes specific $\lambda_{\text{match}}$ thresholds relative to assignment gaps. Theorem 5 assumes natural-image inputs. The conditions are stated explicitly in the theorem statements and proofs; the framework's behavior outside these conditions is empirical, not provable.

**The defense has trade-offs.** The framework imposes $\sim 2\times$ latency overhead, requires training of an auxiliary denoiser, and introduces approximately 7 hyperparameters that must be calibrated to the deployment regime. These trade-offs are explicit and quantifiable, but not zero. A deployment for which the trade-offs are unacceptable is a deployment for which the framework is the wrong defense; we do not claim universal applicability.

## 13. Discussion

### 13.1. The Defense as Detector-Internal Uncertainty Layer

The framework can be characterized at a higher level of abstraction than its specific mathematical formulation. Viewed structurally, it is an *uncertainty layer* that sits between the detector's raw outputs and its decision logic. The suspicion field provides a per-pixel scalar measure of off-manifoldness; the box-pooled coefficient $\beta_i$ provides a per-prediction uncertainty assessment; the four injection mechanisms regulate the detector's confidence, class behavior, localization, and assignment in proportion to that uncertainty.

This perspective situates the framework within the broader literature on uncertainty-aware detection. Modern detectors are generally not uncertainty-aware: they produce confident predictions across the full range of input conditions, including conditions far outside their training distribution. The framework introduces a principled uncertainty signal — grounded in the score function of the natural-image distribution per Theorem 1 — that the detector consumes through differentiable injection. The result is a detector whose confidence is calibrated against the natural-image prior, not just the training distribution.

This characterization clarifies what the framework does and does not provide. It does not provide protection against adversarial inputs that lie *within* the natural-image distribution. It does not provide protection against attacks that operate on the detector's training-time biases. It does provide protection against attacks that produce inputs visibly inconsistent with the natural-image distribution — the characteristic signature of patch attacks, which are produced by optimization against the detector and lie off the natural-image manifold by construction.

The framework is not a universal adversarial defense. It is a defense against the specific structural property that adversarial patches share: they are off-manifold artifacts inserted into otherwise-natural images.

### 13.2. The Hybrid Residual Engine and Information-Theoretic Capture

The framework's design includes both $L_1$ and $L_2$ residual operators, fused into a hybrid mode. Proposition 1 established that the two operators have asymmetric sensitivity profiles. A complementary perspective: the $L_1$ and $L_2$

residuals capture *partially orthogonal information* about off-manifold structure.

The $L_1$ residual captures information about *which* channels are inconsistent with natural-image content. A patch that disrupts a single channel (e.g., a colored adversarial patch in a high-saturation color) produces a $L_1$ residual concentrated on that channel, but spread out by the per-channel averaging into a moderate $L_1$ magnitude.

The $L_2$ residual captures information about *how strongly* the channels are inconsistent. The same single-channel patch produces a strong $L_2$ residual because the squared norm preserves the magnitude of the strongest contribution.

The hybrid mode's fusion $w_1 \cdot L_1 + w_2 \cdot L_2$ weights these complementary captures. The framework does not commit to a fixed information capture; it offers a parameterized family of captures that can be tuned to the deployment's expected attack distribution. A deployment expecting channel-concentrated adversarial patterns should weight $w_2$ higher; a deployment expecting channel-distributed patterns should weight $w_1$ higher; a deployment with uncertain attack distribution should use the neutral $w_1 = w_2 = 0.5$.

### 13.3. Defense in the Edge-Deployment Regime

The framework's edge-budget compliance reflects a deliberate choice about what kind of defense is meaningful for the YOLO26 deployment context. Defenses imposing 100×–300× latency multipliers are not viable in deployments that chose YOLO26 specifically for its 38.9 ms CPU baseline. The framework's ∼2× multiplier is qualitatively different.

The framework demonstrates that a defense can be designed *for* edge deployment, rather than designed in the unconstrained-latency regime and then forced into edge deployment via approximations. The architectural commitments — low-resolution computation, batch-parallel probing, no iterative refinement, single-pass forward flow — are not retrofits to make a slower defense fit edge constraints. They are the framework's foundational design choices.

This stands in contrast to attempts to optimize prior defenses for edge deployment. DiffPure with reduced timesteps trades robustness for speed; PatchCleanser with reduced mask voting trades certification for speed; DIFFender with optimization-based inpainting trades quality for speed. Each represents a defense designed for a different latency regime, compromised to fit the edge regime. The framework's lesson is that the latency barrier is not fundamental to the patch-defense problem; it is fundamental to specific architectural choices made in the prior literature.

### 13.4. The Role of Theorem 5 in the Defense Landscape

We state Theorem 5 explicitly as the framework's fifth and final theoretical result.

**Theorem 5** (No-False-Suppression Under Clean Inputs)**.** *Let $p_0$ be the natural-image distribution and assume (i) the denoiser $\varepsilon_\theta$ is sufficiently close to the optimal denoiser $\varepsilon^*_\theta$ such that the bias term in Theorem 1 is negligible; (ii) calibration constants are chosen per the structural guidance of Section 5.16.3. Then for $x_0 \sim p_0$, the modulated assignment $\pi^*_{DPC}$ matches the unmodulated assignment $\pi^*$ with probability approaching 1 as the suspicion threshold $\tau \to 0$.*

Theorem 5 establishes a structural property no prior empirical diffusion-based defense provides: clean inputs do not have their predictions suppressed by the framework in the limit of low suspicion. This is the *clean-accuracy preservation* property that prior empirical defenses can only validate empirically.

DiffPure's clean-accuracy preservation is empirical: the paper reports that purification at $t^* = 0.075$ preserves classification accuracy within ∼1–2% of baseline on standard datasets, but no theorem guarantees this. DIFFender similarly reports empirical clean-accuracy preservation but does not prove it. PatchCleanser provides certified accuracy under specific conditions but the certification is not the same as preservation: it bounds the accuracy worst-case over patches, not the accuracy on clean inputs.

The framework's structural clean-accuracy guarantee is qualitatively different. It says: under the natural-image distribution and bounded denoiser approximation error, the modulated assignment recovers the unmodulated assignment with probability approaching 1. This is a property that holds for any input drawn from the natural-image distribution. The proof is given in Appendix I.

### 13.5. Future Work

The paper's principal current limitation is that full empirical results under the Section 11 evaluation protocol are pending. The reference codebase has been end-to-end smoke-validated; production multi-seed evaluation is in progress.

**Empirical validation against the Section 11 protocol** is the immediate next step. Production runs across three seeds are scheduled and will produce paper-publishable detection metrics with bootstrap confidence intervals. The framework's structural guarantees provide the theoretical foundation; the empirical numbers will substantiate the deployed claims.

**Adaptive evaluation against specialized attackers** beyond Algorithm 4. The framework should be evaluated against the strongest attacks the community can construct, including attacks specifically designed against assignment-cost modulation as discussed in Section 12.2.

**Empirical verification of Theorem 5 in finite-data regimes.** Theorem 5 holds in the limit of optimal denoiser and natural-image distribution; finite-data behavior on practical training corpora requires empirical verification across multiple datasets.

**Extension to other end-to-end detector families.** Adaptation to DETR-class detectors [19, 20] is possible per the interface specification of Section 7.4. The empirical performance of the framework on these detectors is unknown and warrants investigation.

**Multi-task injection mechanisms in code.** The reference codebase [31] focuses on detection. The four remaining task-specific injection mechanisms (segmentation, pose, OBB, classification) are mechanically derivable per Section 6.2 and are pending in v3.3.x extensions of the reference codebase.

**Compositional defenses.** The framework's interaction with image-space pre-processors — composing detector-internal injection with image-space defenses — is an underexplored space that may yield empirical robustness exceeding either component alone.

### 13.6. Threat Model Boundary Cases

We articulate explicitly what the framework does and does not defend against. Each case is stated as the response to a specific adversary capability or input distribution.

**Patches drawn from the natural-image distribution.** The framework's structural argument rests on adversarial patches being off-manifold artifacts produced by optimization against a model. Patches that lie within the natural-image distribution — for example, real-world objects that happen to confuse the detector through their natural appearance — are not off-manifold by construction and would not register strongly in the suspicion field. The framework offers no protection against this class of inputs. They constitute a different problem (out-of-distribution detection or confusing-but-real objects) that the framework does not claim to address.

**Adversarial patterns embedded in semantically-natural content.** An attacker who learns the framework's score-function approximation can in principle construct patches that lie close to the natural-image manifold while still degrading the detector. Such patches would produce low suspicion signal and could evade the framework's intervention. The framework's defense against this case relies on the conjecture that such patches require substantially more attack budget than off-manifold patches; empirical validation of this conjecture against specifically-constructed manifold-preserving attacks is required and is not yet provided.

**Distribution shift between training and evaluation.** The denoiser is trained on a specific natural-image corpus. If the evaluation domain differs substantially from the training corpus — different illumination conditions, different sensor characteristics, different scene compositions — the denoiser's residual signal on clean inputs will increase, eroding Theorem 5's clean-accuracy guarantee. The framework provides no automatic adaptation to distribution shift; deployments must retrain or fine-tune the denoiser on representative target-domain data.

**Attacks that target the framework's hyperparameters.** An attacker with white-box access to $\lambda_{\text{obj}}, \lambda_{\text{cls}}, \lambda_{\text{match}}$ can in principle construct patches that exploit specific calibration choices. The framework's structural guarantees of Theorem 3 apply under stated $\lambda_{\text{match}}$ conditions; attackers that operate below those conditions face the structural guarantee, attackers that operate above them face a different empirical regime. The framework's calibration constants should not be communicated as cryptographic secrets, but the deployment's chosen values should be selected with awareness that they define the structural-guarantee envelope.

**Patches occupying a substantial fraction of the image.** The framework is designed for patches occupying $\rho \leq 5\%$ of the image area. Patches occupying larger fractions ($\rho > 20\%$, for example) violate the localized-perturbation assumption that motivates the box-pooling approach. Such patches are not the canonical patch attack of Brown et al. [6] and prior literature; they constitute a different class of attack (large-area image corruption) that this framework does not target.

**Coordinated multi-patch attacks.** An attacker who positions multiple patches simultaneously may evade the assignment-cost modulation if the suspect set $R$ exceeds the framework's redirect capacity. The framework's behavior in this case is graceful degradation: the modulated cost matrix retains its mathematical structure, but the protective margin established by Theorem 3 property (b) may be insufficient to redirect all matches. Multi-patch attacks are not specifically analyzed in this paper; preliminary expectation is that the framework's effectiveness will decline approximately linearly in the number of patches up to the point where the suspect set covers a substantial fraction of the image.

**Attacks that exploit the denoiser's training data.** If the denoiser's training data is itself adversarially compromised (a data-poisoning attack on the framework's training pipeline), the resulting denoiser will produce systematically biased residual signal. The framework provides no defense against compromise of its training pipeline; standard data-pipeline-integrity controls apply.

Articulating these boundaries serves two purposes. First, it prevents over-claiming: the framework is a defense against a specific class of attacks and not a universal adversarial defense. Second, it identifies the directions in which subsequent work can extend the framework's coverage — each boundary case is a target for future research, not a permanent limitation.

## 14. Conclusion

This paper has introduced a defense designed for the architectural specifics of NMS-free end-to-end detectors, exemplified by YOLO26. The defense addresses three attack surfaces that the prior patch-defense literature has not engaged with: the assignment-matrix attack surface created by learned end-to-end matching, the multi-task head attack surface created by unified backbones, and the edge-latency constraint created by deployment-oriented detector design.

The framework's distinguishing architectural commitment is detector-internal injection. Rather than transforming the input image and feeding the result to an unmodified detector — the paradigm of every prior diffusion-based and entropy-based patch defense — the framework computes a spatial suspicion field and injects it directly into the detector's decision logic at four points: objectness calibration, class-logit suppression, localization stabilization, and modulation of the Hun-

garian matching cost matrix. The fourth point, the assignment-cost modulation, is the framework's most distinctive contribution: it directly addresses an attack surface that exists only in NMS-free end-to-end detection and that no prior defense can reach.

The framework rests on five theorems giving it theoretical backbone. Theorem 1 establishes that the per-pixel residual signal is an asymptotically unbiased estimator of the natural-image score function under the optimal denoiser. Theorem 2 bounds the variance of the $K$-probe ensemble field. Theorem 3, the central theoretical contribution, characterizes the assignment-cost modulation as recovering clean assignments, redirecting suspect assignments away from off-manifold regions, and behaving smoothly under perturbation. Theorem 4 bounds end-to-end latency at twice the baseline detector cost. Theorem 5 establishes the structural clean-accuracy preservation property.

The framework is accompanied by an open-source reference implementation [31], release v3.3.1, that operationalizes Theorems 1 through 3 in deployable form. The implementation has been validated end-to-end via a 13.3-minute smoke run that exercises every component of the pipeline on Apple Silicon hardware. The smoke run confirms that the modulated Hungarian assignment of Theorem 3 / Equation (30) fires correctly on real COCO ground-truth labels at runtime — the central empirical signal that the math layer and the detector layer are correctly coupled in code. Production multi-seed runs against the Section 11 evaluation protocol are scheduled following submission and will produce the framework's first paper-publishable detection metrics with bootstrap confidence intervals.

The broader lesson, beyond the specific framework, is that detector architectures and their defenses are co-evolving. As detector architectures shift toward end-to-end design, multi-task unification, and edge-deployment constraints, the appropriate defense architectures must shift correspondingly. The patch-defense literature's architectural consistency reflects the architectural consistency of the systems that have been the focus of attention — primarily classifiers and NMS-based detectors. This paper contributes one example of the architectural shift in defense that NMS-free end-to-end detection requires, and articulates the broader research direction within which subsequent contributions will develop.

## 14.5. Practical Deployment — Reference Codebase v3.3.1

This section provides the operational counterpart to the framework's mathematical specification. We describe the open-source reference implementation [31] of the framework, document the end-to-end smoke-validation evidence from May 12, 2026, and give three practical workflows: pipeline validation via the smoke configuration, production multi-seed runs, and library-level use of the defense as a wrapped detector.

### *14.5.1. Reference Codebase Overview*

The reference codebase implements the framework as a three-layer system. Every theorem-relevant mathematical operation is realized as a pure-function module in a math library; every training and evaluation procedure is a command-line tool that composes those modules; and an orchestration plane manages the eight-phase pipeline with multi-seed reproducibility and integrity manifests.

**Table 2:** Reference codebase v3.3.1 layer decomposition.

| Layer | Contents |
|---|---|
| Math library (`dpc/`) | 22 pure-function PyTorch modules. Each theorem-relevant function carries a docstring tag of form "Implements §X / Eq. (N)" pointing to this paper. Key modules: `assignment.py` (Eq. 30, Thm. 3), `calibration.py` (Eqs. 23, 24, 29), `field.py` ($K$-probe engine, §5), `pooling.py` (Eq. 22), `wrapper.py` (DPCWrapper), `yolo26_native.py` (YOLO26 head bridge), `denoiser.py` (TinyUNet). |
| Training/eval tools (`tools/`) | Command-line scripts implementing each phase: `build_caches.py`, `train_phase1.py`, `train_phase2.py`, `evaluate_phase3.py`, `diagnose_residuals.py`, `fit_color_distribution.py`, `compare_phases.py`. |
| Orchestrator (`dpcctl/`) | Control plane: `cli.py`, `config.py` (JSON schema), `orchestrator.py` (phase scheduler), `events.py`, `dashboard.py`, `paths.py`, one `Phase` subclass per pipeline step under `phases/`. |
| Test suite (`tests/`) | 107 unit tests covering every theorem-relevant function. Theorem 3 properties (a), (b), (c) tested directly via constructed examples. Full suite runs in approximately 30 seconds. |
| Configurations (`configs/`) | Three presets: `quick.json` (13-minute smoke), `default.json` (1-hour development), `production.json` (3 seeds × 20–24 hours). |

The reference codebase ships with a complete user manual, a design document mapping every source file to its responsibility, a changelog tracking version-over-version differences, and per-tool argparse help text fully describing each command-line flag.

### *14.5.2. Smoke-Validation Evidence (May 12, 2026)*

The reference codebase was validated end-to-end on May 12, 2026 against the quick configuration on a MacBook Pro M1 Max running macOS 14, Python 3.12.13, PyTorch 2.11 with Apple MPS backend. The full eight-phase pipeline completed in 13.3 minutes wall-clock.

**Table 3:** Smoke test phase-by-phase results, May 12 2026, M1 Max MPS backend, reference codebase v3.3.1.

| Phase | Time | Outcome | Key observation |
|---|---|---|---|
| `prep` | 3.7 min | PASS | 4 caches, 24.57 GB, all SHA256s valid |
| `train_p1` | 2.4 min | PASS | Val loss 0.92 → 0.59 over 2 epochs |
| `diagnose_p1` | 1.4 min | PASS | 873 images, median ratio 1.002 (CI 95% [1.000, 1.004]) |
| `train_p2` | 7.0 min | PASS | `n_matched` range $[3, 60]$ per batch — Eq. 30 firing |
| `diagnose_p2` | 1.4 min | PASS | median ratio 0.996; phase 1 vs phase 2 delta computed |
| `eval_p3` | 0.7 min | PASS | 5 alphas × 50 images; pipeline ran |
| `eval_negative_control` | 0.4 min | PASS | 30 clean images; pipeline ran |
| `aggregate` | $<0.1$ min | PASS | `alpha_sweep.json` produced |

The most consequential single observation from the smoke run is the `n_matched` range during Phase 2 joint training. The Hungarian assignment of Theorem 3 / Equation 30 was observed to match 3 to 60 (prediction, ground-truth) pairs per batch on real COCO data, confirming that the modulated assignment is operating non-trivially on production ground-truth labels and not on synthetic placeholder data. A value of zero across many consecutive steps would have indicated that ground truth was not flowing through the pipeline—a failure mode that does not arise in the validated v3.3.1 release.

Phase 3 evaluation in the smoke configuration produces degenerate zero-detection results by design: the smoke configuration uses 200 steps per epoch and 2 epochs, which is insufficient for the YOLO26 head to learn the calibration regime. The smoke validates pipeline correctness, not defense effectiveness. Production-scale results require the full configuration described in §14.5.3.

#### *14.5.3. Production Multi-Seed Runs*

The production configuration runs three seeds (42, 1337, 2718) sequentially with full-scale training and the complete APRICOT evaluation set. Wall-clock is approximately 20–24 hours on a single M1 Max. Each seed's outputs land under `runs_production_v33/production/seed_N/`, with aggregate cross-seed statistics in `aggregate/aggregate_across_seeds.json`.

The standard production workflow is two commands:

```
# 1. Validate the configuration
python -m dpcctl validate -c configs/production.json

# 2. Run the full pipeline
python -m dpcctl run -c configs/production.json -p all

# Output: runs_production_v33/production/aggregate/aggregate_across_seeds.json
```

The aggregate JSON contains the framework's three primary defense metrics with bootstrap 95% confidence intervals across seeds: `on_patch_suppression` (fraction of patch-region false positives suppressed), `off_patch_retention` (fraction of true positives away from the patch preserved), and `per_image_margin` (per-image difference between DPC and baseline). These correspond to the assignment-matrix attack metrics enumerated in §11.

The orchestrator persists per-phase state under `runs_*/seed_*/<phase>/<phase>_state.json`. If a run is interrupted, re-invocation skips already-completed phases automatically, resuming from the next unstarted phase. The `--force` flag overrides this for deliberate re-execution.

#### *14.5.4. Library API — DPCWrapper*

For deployment scenarios where the framework is consumed as a library component rather than a full training pipeline, the reference codebase exposes the trained defense as a single `nn.Module`. Given a Phase 1 denoiser checkpoint and a Phase 2 fine-tuned YOLO26 head, the user can construct a DPCWrapper that takes images and returns calibrated detections:

```
import torch
from dpc.wrapper import DPCWrapper
from dpc.denoiser import TinyUNetDenoiser
from dpc.config import DPCConfig
from dpc.yolo26_native import load_yolo26, patch_finetuned_head

cfg = DPCConfig()                # paper defaults: K=8, h=w=128, hybrid, sigma_smooth=1.5
device = torch.device('mps')     # or 'cuda', or 'cpu'

# Load Phase 1 EMA denoiser checkpoint
denoiser = TinyUNetDenoiser().to(device)
denoiser.load_state_dict(torch.load(
    'runs_production_v33/production/seed_42/train_p2/checkpoints/latest/ema.pt'))

# Load YOLO26, patch the fine-tuned head from Phase 2
yolo26 = load_yolo26('yolo26n.pt', device)
patch_finetuned_head(yolo26,
    'runs_production_v33/production/seed_42/train_p2/yolo26_head_finetuned.pt')

# Wrap and use
defense = DPCWrapper(yolo26, denoiser, cfg).to(device).eval()
image = torch.rand(1, 3, 320, 320, device=device)  # batch of one
detections = defense(image)
# detections.boxes_xyxy, .scores, .classes, .beta (per-anchor suspicion)
```

The DPCWrapper implementation directly composes the math library: the Brownian suspicion engine (Layer 2 of §7) is `dpc.field.DPCField`; box pooling (Layer 4) is `dpc.pooling.box_pool_grid`; the four calibration

injections (Layer 5) are `dpc.calibration.calibrate_predictions`, which detects the YOLO26-specific absence of a separate objectness logit and routes through the collapsed $(\lambda_{\text{obj}} + \lambda_{\text{cls}})$ class shift described in §5. All operations in DPCWrapper are differentiable end-to-end, satisfying the gradient-flow requirement for adaptive evaluation.

#### *14.5.5. Reproducibility*

Within one seed, every random decision in the reference codebase is deterministic: `torch.manual_seed`, `numpy.random.seed`, Python `random.seed` all set; DataLoader `worker_init_fn` seeds each worker; $K$-probe RNG seeded per step; COCO train/val split via `deterministic_split`; synthetic patch generator and mixture loader both seeded by `(base_seed` $\times$ `prime` $+$ `i)` mod $2^{31}$. EMA, optimizer state, scheduler state, and RNG state are all checkpointed atomically. Resume from checkpoint produces numerically identical output to a single uninterrupted run.

Each run writes a manifest containing SHA256 of every output file plus the environment fingerprint (Python version, library versions, hardware identifier, fully-resolved configuration). Re-running with the same seed and the same manifest reproduces the run exactly.

Cross-seed variance is the bootstrap signal used by the aggregate phase. The three production seeds (42, 1337, 2718) provide three independent estimates of every framework metric; the aggregate phase computes 95% bootstrap confidence intervals across these estimates and writes the result to `aggregate_across_seeds.json`.

#### *14.5.6. Where the Theorems Live in Code*

For readers wishing to navigate from the paper's theoretical results to their concrete implementation:

**Table 4:** Map from paper results to v3.3.1 source files.

| **Paper result** | **Code location** |
|---|---|
| Theorem 1 (Residual $\approx$ score) | `dpc/field.py::compute_field`. $K$-probe loop at `field.py:87`, residual at `field.py:140` ($L_1$) and `field.py:155` ($L_2$). |
| Theorem 2 (Variance reduction) | `dpc/field.py`. $K = 8$ is `cfg.n_probes` default; independence enforced by per-probe noise sampling at `field.py:120`. |
| Theorem 3 (Modulated assignment) | `dpc/assignment.py`. `modulate_cost_matrix` implements Eq. 30; `hungarian_assign` wraps `scipy.optimize.linear_sum_assignment`. Properties (a), (b), (c) directly tested in `tests/test_assignment.py`. |
| Theorem 4 (Latency bound) | Verified during production-scale benchmarking; reported in `CHANGELOG.md`. |
| Theorem 5 (Clean accuracy) | Operational consequence of Theorem 3(a). `dpc/calibration.py::calibrate_predictions` reduces to identity when $\beta = 0$. |
| Proposition 1 ($L_1/L_2$ asymmetry) | Computed branches at `field.py:140-160`. Hybrid fusion at `field.py:168`. |
| Proposition 2 (Box pooling continuity) | `dpc/pooling.py::box_pool_grid`. Bilinear sampling at `pooling.py:38`. |
| Proposition 3 (Multi-task) | Detection task: `dpc/calibration.py + dpc/wrapper.py`. Multi-task extensions specified in §6.2 and pending in v3.3.x extensions. |
| Equation 23 (Objectness calibration) | `dpc/calibration.py::calibrate_objectness`. |
| Equation 24 (Class calibration) | `dpc/calibration.py::calibrate_class_uniform`. |
| Equation 27 (Class-entropy regularizer) | `dpc/auxiliary_losses.py::class_entropy_regularizer`. |
| Equation 28 (Box stability) | `dpc/auxiliary_losses.py::box_stability_loss`. |
| Equation 29 (Small-target amplification) | `dpc/calibration.py::amplify_small_targets`. |
| Equation 30 (Modulated cost) | `dpc/assignment.py::modulate_cost_matrix`. |

## Acknowledgments

The author gratefully thanks Dr. Yanyan Li and Dr. Joshua Harguess for their support and guidance throughout the development of this work. This work was conducted at California State University San Marcos.

## References

[1] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Comput.*, vol. 23, no. 7, pp. 1661–1674, Jul. 2011.

[2] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *Proc. ICLR*, 2021.

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. NeurIPS*, vol. 33, 2020, pp. 6840–6851.

[4] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Proc. ICLR*, 2021.

[5] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Proc. NeurIPS*, vol. 32, 2019.

[6] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," arXiv:1712.09665, 2017.

[7] Y.-C.-T. Hu *et al.*, "Naturalistic physical adversarial patch for object detectors," in *Proc. ICCV*, 2021.

[8] S. Thys, W. Van Ranst, and T. Goedemé, "Fooling automated surveillance cameras," in *Proc. CVPRW*, 2019.

[9] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proc. ICML*, 2018.

[10] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security," in *Proc. ICML*, 2018.

[11] F. Tramèr, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," in *Proc. NeurIPS*, 2020.

[12] N. Carlini *et al.*, "On evaluating adversarial robustness," arXiv:1902.06705, 2019.

[13] C. Xiang, S. Mahloujifar, and P. Mittal, "PatchCleanser: Certifiably robust defense against adversarial patches," in *Proc. USENIX Security*, 2022.

[14] C. Xiang, A. Valtchanov, S. Mahloujifar, and P. Mittal, "ObjectSeeker: Certifiably robust object detection against patch hiding attacks," in *Proc. IEEE S&P*, 2023.

[15] J. Liu, A. Levine, C. P. Lau, R. Chellappa, and S. Feizi, "Segment and complete: Defending object detectors," in *Proc. CVPR*, 2022.

[16] B. Tarchoun, A. Ben Khalifa, M. A. Mahjoub, N. Abu-Ghazaleh, and I. Alouani, "Jedi: Entropy-based localization and removal of adversarial patches," in *Proc. CVPR*, 2023.

[17] W. Nie *et al.*, "Diffusion models for adversarial purification," in *Proc. ICML*, 2022.

[18] C. Kang, Y. Dong, Z. Wang, S. Ruan, Y. Chen, H. Su, and X. Wei, "DIFFender: Diffusion-based adversarial defense against patch attacks," in *Proc. ECCV*, 2024.

[19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. ECCV*, 2020.

[20] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR," in *Proc. ICLR*, 2021.

[21] R. Sapkota, R. H. Cheppally, A. Sharda, and M. Karkee, "YOLO26: Key architectural enhancements and performance benchmarking for real-time object detection," arXiv:2509.25164, 2025.

[22] R. Sapkota and M. Karkee, "Ultralytics YOLO evolution: An overview of YOLO26, YOLO11, YOLOv8 and YOLOv5," arXiv:2510.09653, 2025.

[23] G. Jocher and J. Qiu, "Ultralytics YOLO26," 2026. [Online]. Available: https://docs.ultralytics.com

[24] S. M. Turjya, "A comprehensive review of YOLOv26 and existing YOLO variants," 2026.

[25] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN," in *Proc. ICLR*, 2018.

[26] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "PixelDefend," in *Proc. ICLR*, 2018.

[27] A. Braunegg *et al.*, "APRICOT: A dataset of physical adversarial attacks on object detection," in *Proc. ECCV*, 2020.

[28] M. Threet *et al.*, "Physical adversarial attacks in simulated environments," in *Proc. IEEE AIPR*, MITRE Corporation, 2021.

[29] H. Robbins, "An empirical Bayes approach to statistics," in *Proc. 3rd Berkeley Symp.*, 1956.

[30] B. Efron, "Tweedie's formula and selection bias," *J. Am. Stat. Assoc.*, vol. 106, no. 496, pp. 1602–1614, 2011.

[31] C. Varela, "DPC-YOLO26: Diffusion-prior consistency defense for NMS-free detection — reference implementation, release v3.3.1," California State University San Marcos, May 2026.

## A. Notation

The notation conventions follow Song et al. [2] for the diffusion forward process and Vincent [1] for the score-denoising connection.

**Diffusion forward process**

| | |
|---|---|
| $x_0$ | Clean input image in $[0,1]^{C\times H\times W}$ |
| $x_t$ | Forward-perturbed image at $t \in [\varepsilon_{\min}, 1]$ |
| $t_k, K$ | Probe timesteps and count ($K = 8$) |
| $\alpha(t), \sigma(t)$ | Signal retention and noise magnitude |
| $\varepsilon_k \sim \mathcal{N}(0, I)$ | Independent noise at probe $k$ |
| $\varepsilon_\theta, \varepsilon_\theta^*$ | Trained and optimal denoisers |
| $p_t, p_0$ | Forward marginal and natural-image distribution |

**Suspicion engine**

| | |
|---|---|
| $h \times w, H \times W$ | Probe resolution (128) and image resolution |
| $\Delta_k(u,v) \in \mathbb{R}^C$ | Per-probe residual |
| $r_k^{(L_1)}, r_k^{(L_2)}$ | $L_1$ and $L_2$ residual maps |
| $G_{\sigma_{\text{smooth}}}$ | Gaussian smoothing kernel ($\sigma = 1.5$) |
| $R_m, \hat{R}_m$ | Branch and normalized branch field |
| $R_{\text{hyb}}, \hat{R}_{\text{hyb}}$ | Hybrid fused field |
| $R_{\text{deploy}}, I_{\text{img}}$ | Deployed field; upsampled to image |
| $w_1, w_2$ | Fusion weights, $w_1 + w_2 = 1$ |

**Detector predictions and assignment**

| | |
|---|---|
| $\hat{b}_i, s_i, z_i$ | Predicted box, objectness logit, class logits |
| $\beta_i, \beta_i^{(\text{small})}$ | Box-pooled and amplified suspicion |
| $P$ | ROI sample grid size ($P = 7$) |
| $\pi, \pi^*, \pi^*_{\text{DPC}}$ | Assignment, optimal, modulated optimal |
| $C_{\text{base}}, C_{\text{DPC}}$ | Base and modulated cost matrix |
| $\gamma_k(R), \gamma_{\min}$ | Assignment gap; minimum gap |

**Calibration constants**

| | |
|---|---|
| $\lambda_{\text{obj}}, \lambda_{\text{cls}}, \lambda_{\text{match}}$ | Calibration weights (Eqs. 23, 24, 30) |
| $\lambda_{\text{small}}, a_{\min}$ | Small-target amplification |
| $\lambda_{\text{cls-prior}}, \lambda_{\text{box-stab}}$ | Loss weights |
| $\alpha, B, T_{\text{iter}}$ | Adaptive attack parameters |

# B. Recommended Experiment Table Template

This appendix provides recommended table templates for any experimental work building on the framework. The templates correspond to the evaluation modes and attack surfaces specified in Section 11. They are intended as a starting point for empirical reporting, not as a rigid format.

## B.1. Detection Results Template (mAP, %)

**Table 5:** Detection mAP under clean and patch-attacked inputs, three evaluation modes.

| Configuration | Clean $\text{mAP}_{50:95}$ | Attacked $\text{mAP}_{50:95}$ | Δ vs M |
|---|---|---|---|
| Mode A (baseline) | xx.x ± y.y | xx.x ± y.y | |
| Mode B (DPC defense) | xx.x ± y.y | xx.x ± y.y | + |
| Mode C (denoise+detect) | xx.x ± y.y | xx.x ± y.y | + |
| DiffPure [17] | xx.x ± y.y | xx.x ± y.y | + |
| DIFFender [18] | xx.x ± y.y | xx.x ± y.y | + |
| Jedi [16] | xx.x ± y.y | xx.x ± y.y | + |

## B.2. Multi-Task Coverage Template

**Table 6:** Per-task metrics under attack; Mode B (defended) vs Mode A (baseline).

| Task | Metric | Mode A | Mode B |
|---|---|---|---|
| Detection | $\text{mAP}_{50:95}$ | xx.x | xx.x |
| Segmentation | Mask $\text{mAP}_{50:95}$ | xx.x | xx.x |
| Pose | Keypoint $\text{mAP}_{50:95}$ | xx.x | xx.x |
| OBB | $\text{mAP}_{50:95}$ | xx.x | xx.x |
| Classification | Top-1 (%) | xx.x | xx.x |

## B.3. Latency Template

**Table 7:** Per-image latency, target hardware = (specify platform).

| Configuration | Median (ms) | p95 (ms) | Mult. vs A |
|---|---|---|---|
| Mode A (baseline) | xx.x | xx.x | 1.00 |
| Mode B (DPC defense) | xx.x | xx.x | x.xx |
| Mode C (denoise+detect) | xx.x | xx.x | x.xx |
| DiffPure (default) | xx.x | xx.x | xx.xx |
| DIFFender (optimized) | xx.x | xx.x | xx.xx |
| PatchCleanser (default) | xx.x | xx.x | xx.xx |

## B.4. Assignment-Surface Metrics Template

**Table 8:** Direct measurements of the three assignment-matrix attack classes.

| Metric | Mode A | Mode B |
|---|---|---|
| Hide rate (%, lower = better) | xx.x | xx.x |
| Class redirection rate (%, lower) | xx.x | xx.x |
| False positives per image (lower) | x.xx | x.xx |
| Assignment displacement (%, higher = better redirect) | xx.x | xx.x |

### B.5. Diagnostic Conditions Template

**Table 9:** Athalye-Carlini-Wagner [10] diagnostic conditions for the framework.

| Diagnostic | Outcome |
|---|---|
| (1) PGD-1000 success > FGSM success | yes / no |
| (2) White-box success > transfer success | yes / no |
| (3) Unbounded budget reaches 100% | xx.x % at $\rho = 0.5$ |
| (4) Brute-force $\leq$ PGD success | yes / no |
| (5) Monotonic in patch size | yes / no |

A defense reporting these five diagnostics with positive outcomes on all five rows satisfies the obfuscated-gradient screen of Athalye, Carlini, and Wagner [10].

## C. Proof of Theorem 1

We restate Theorem 1. Let $\varepsilon_\theta^*$ be the optimal denoiser minimizing the score-matching objective, and let $p_t$ denote the marginal distribution of $x_t$ under the forward VP-SDE. Then for any clean input $x_0$ and probe timestep $t_k$,

$$\mathbb{E}_{\varepsilon_k}\left[\|\Delta_k(u,v)\|_2^2 \mid x_0\right] = \sigma(t_k)^2\,\mathbb{E}_{\varepsilon_k}\left[\|\nabla_{x_{t_k}} \log p_{t_k}(x_{t_k})(u,v)\|_2^2 \mid x_0\right] + C_{u,v}.$$

**Foundational identities**

**Lemma 1** (Vincent's Score-Denoising Identity [1])**.** *Under the VP-SDE forward process with marginal distribution $p_t$, the optimal denoiser satisfies $\varepsilon_\theta^*(x_t,t) = -\sigma(t)\nabla_{x_t}\log p_t(x_t)$ for almost every $(x_t,t)$.*

**Lemma 2** (Tweedie's Formula in VP-SDE Form [29, 30])**.** *For the marginal $p_t$:*

$$\mathbb{E}_{x_0\sim p_0}[x_0 \mid x_t] = \frac{x_t + \sigma(t)^2\nabla_{x_t}\log p_t(x_t)}{\sqrt{\alpha(t)}}.$$

**Proof of Theorem 1**

Fix $x_0$ and probe timestep $t_k$. Let $\varepsilon \sim \mathcal{N}(0,I)$ and $x_{t_k} = \sqrt{\alpha(t_k)}x_0 + \sigma(t_k)\varepsilon$. The per-pixel residual at $(u,v)$ is

$$\Delta_k(u,v) = \varepsilon_\theta^*(x_{t_k},t_k)(u,v) - \varepsilon(u,v).$$

Substituting Lemma 1,

$$\Delta_k(u,v) = -\sigma(t_k)\nabla_{x_{t_k}}\log p_{t_k}(x_{t_k})(u,v) - \varepsilon(u,v).$$

Taking the squared $L_2$ norm,

$$\begin{aligned}\|\Delta_k(u,v)\|^2 = {}& \sigma(t_k)^2\,\|\nabla\log p_{t_k}(x_{t_k})(u,v)\|^2 \\ &+ 2\sigma(t_k)\,\langle\nabla\log p_{t_k}(x_{t_k})(u,v), \varepsilon(u,v)\rangle \\ &+ \|\varepsilon(u,v)\|^2.\end{aligned}$$

**Term 1.** $\sigma(t_k)^2\cdot\|\nabla\log p_{t_k}\|^2$, the score-function term.

**Term 2.** The cross-term. The Gaussian conditional $q(x_{t_k}\mid x_0)$ has score with respect to $x_{t_k}(u,v)$:

$$\nabla_{x_{t_k}(u,v)}\log q(x_{t_k}\mid x_0) = -\varepsilon(u,v)/\sigma(t_k).$$

Thus $\varepsilon(u,v) = -\sigma(t_k)\nabla_{x_{t_k}(u,v)}\log q$. By Stein's integration-by-parts identity, for any $f$ of $x_{t_k}$:

$$\mathbb{E}[f(x_{t_k})\nabla\log q(x_{t_k}\mid x_0)\mid x_0] = -\mathbb{E}[\nabla f(x_{t_k})\mid x_0].$$

Applying with $f = \nabla\log p_{t_k}(x_{t_k})(u,v)$:

$$\mathbb{E}_\varepsilon[\langle\nabla\log p_{t_k},\varepsilon\rangle\mid x_0] = \sigma(t_k)\,\mathbb{E}[\nabla^2_{x_{t_k}(u,v)}\log p_{t_k}\mid x_0].$$

**This cross-term does not vanish in general.** It equals $\sigma(t_k)$ times a Hessian-trace expectation that depends on $x_0$ through the local curvature of $\log p_{t_k}$.

**Limitation L1 (Conditional Form).** Theorem 1 in the conditional form requires either (a) replacing the conditional expectation with an unconditional expectation over $(x_0,\varepsilon)$ jointly, in which case the Hessian-trace term contributes a quantity independent of any specific $x_0$; or (b) invoking a local-regularity assumption that the Hessian of $\log p_{t_k}$ varies slowly across the support of $x_{t_k}\mid x_0$, so the Hessian-trace contribution is approximately constant in $x_0$. In well-trained denoiser regimes on natural images, assumption (b) is reasonable.

Under either interpretation, the cross-term contributes $C_{\text{cross}}$ — constant (case a) or approximately constant (case b).

**Term 3.** $\mathbb{E}[\|\varepsilon(u,v)\|^2] = C$ (the channel count), independent of $x_0$.

Combining the three terms,

$$\mathbb{E}_\varepsilon[\|\Delta_k(u,v)\|^2\mid x_0] = \sigma(t_k)^2\,\mathbb{E}[\|\nabla\log p_{t_k}\|^2\mid x_0] + C_{\text{cross}} + C.$$

Setting $C_{u,v} = C_{\text{cross}} + C$ (independent of $x_0$ under the regularity assumption), the theorem statement holds. □

The proof reveals that the cross-term cancellation we initially asserted in the §5 sketch requires either an unconditional-expectation interpretation or a regularity assumption on the local Hessian. The framework's structural argument relies on the expected residual being monotonically related to off-manifoldness, which follows from the proof under either interpretation. The specific magnitude of $C_{u,v}$ does not affect the framework's behavior — calibration constants are tuned empirically.

## D. Proof of Theorem 2

We restate: $\mathrm{Var}[\bar r_K(u,v)\mid x_0] = (1/K^2)\sum_{k=1}^K V_k(u,v) \leq V_{\max}(u,v)/K$.

**Proof**

The $K$-probe ensemble residual is $\bar r_K(u,v) = (1/K)\sum_{k=1}^K X_k(u,v)$, where $X_k(u,v) = \|\Delta_k(u,v)\|^2$.

**Independence verification.** Each $X_k(u,v)$ depends on $(x_0,\varepsilon_k)$. Since $\varepsilon_1,\ldots,\varepsilon_K$ are independent draws and $x_0$ is fixed conditional on the input, the $\{X_k(u,v)\}_{k=1}^K$ are mutually independent random variables conditional on $x_0$. The denoiser's deterministic forward pass cannot introduce cross-probe dependence.

**Variance of the average.** For independent variables,

$$\mathrm{Var}\left[\frac{1}{K}\sum_k X_k\right] = \frac{1}{K^2}\sum_k \mathrm{Var}[X_k] = \frac{1}{K^2}\sum_k V_k(u,v).$$

This establishes the equality.

**Upper bound.** Since $V_k(u,v) \le V_{\max}(u,v)$:

$$\frac{1}{K^2}\sum_k V_k \le \frac{1}{K^2}\cdot K \cdot V_{\max} = \frac{V_{\max}}{K}.$$

This establishes the upper bound. □

**Tightness.** The upper bound $V_{\max}/K$ is tight when $V_k = V_{\max}$ for all $k$. When per-probe variances differ, the bound is loose by a factor up to $K$. In practice, probes at smaller $\sigma(t_k)$ generally have lower variance. The variance reduction rate $O(1/K)$ is sharp asymptotically, justifying $K = 8$ as a quantitative trade-off.

## E. Proof of Proposition 1

**Proof of (a) — concentrated case**

$\Delta$ has a single nonzero entry $\Delta_1 = m$ with $\Delta_c = 0$ for $c \ge 2$.

$$r^{(L_1)} = \frac{1}{C}\sum_c |\Delta_c| = \frac{m}{C}, \quad r^{(L_2)} = \sqrt{\sum_c \Delta_c^2} = m.$$

Ratio: $r^{(L_2)}/r^{(L_1)} = m/(m/C) = C$.

**Proof of (b) — uniform case**

$\Delta$ has all entries equal to $m$.

$$r^{(L_1)} = \frac{Cm}{C} = m, \quad r^{(L_2)} = \sqrt{Cm^2} = m\sqrt{C}.$$

Ratio: $r^{(L_2)}/r^{(L_1)} = \sqrt{C}$. □

**Generalization to $p$-norms**

For $p \ge 1$, the ratio of normalized $p$-norm residuals between concentrated and distributed cases is $C^{1/p-1}$ (concentrated) versus $C^{1/p-1}\cdot C^{1/2}$ (distributed). Larger $p$ emphasizes concentrated structures more strongly.

## F. Proof of Proposition 2

**Proof**

Let $\hat{b} = (x_1, y_1, x_2, y_2)$ and $\hat{b}' = (x_1+\delta_1, y_1+\delta_2, x_2+\delta_3, y_2+\delta_4)$ with $|\delta_i| \le \varepsilon$. Without loss of generality, the perturbed box overlaps the original.

Let $A = A(\hat{b})$ and $A' = A(\hat{b}')$. Decomposing the integration domains $\hat{b} = (\hat{b}\cap\hat{b}')\cup(\hat{b}\setminus\hat{b}')$ and $\hat{b}' = (\hat{b}\cap\hat{b}')\cup(\hat{b}'\setminus\hat{b})$:

$$\beta(\hat{b}) - \beta(\hat{b}') = \left(\frac{1}{A} - \frac{1}{A'}\right)\iint_{\hat{b}\cap\hat{b}'} I_{\text{img}} + \frac{1}{A}\iint_{\hat{b}\setminus\hat{b}'} I_{\text{img}} - \frac{1}{A'}\iint_{\hat{b}'\setminus\hat{b}} I_{\text{img}}.$$

Using $\|I_{\text{img}}\|_\infty \le 1$:

$$|\beta(\hat{b}) - \beta(\hat{b}')| \le \frac{|A'-A|}{AA'}\cdot|\hat{b}\cap\hat{b}'| + \frac{|\hat{b}\setminus\hat{b}'|}{A} + \frac{|\hat{b}'\setminus\hat{b}|}{A'}.$$

The first term is $O(\varepsilon\cdot\text{perimeter}/A)$; the remaining terms are each $\varepsilon\cdot O(\text{perimeter})/A$.

When $I_{\text{img}}$ is $L$-Lipschitz, refining by replacing $\|I_{\text{img}}\|_\infty$ with the $L\cdot\varepsilon$ bound on small affected regions gives $|\beta(\hat{b}) - \beta(\hat{b}')| \le L\cdot\varepsilon\cdot\kappa$, where $\kappa$ is bounded by a small multiple of 1. □

**Implication.** The Lipschitz constant $L$ of $I_{\text{img}}$ is bounded by the smoothing operator's effect: Gaussian smoothing with bandwidth $\sigma_{\text{smooth}}$ produces a field with Lipschitz constant $O(1/\sigma_{\text{smooth}})$.

## G. Proof of Theorem 3

**Proof of Property (a) — Recovery**

If $\beta = 0$, then $C_{\text{DPC}}(i,k) = C_{\text{base}}(i,k)$ for all $(i,k)$. Therefore $C_{\text{DPC}} = C_{\text{base}}$ and the optimal assignments coincide: $\pi^*_{\text{DPC}} = \arg\min_\pi \sum_k C_{\text{DPC}}(\pi(k),k) = \pi^*$. □

**Proof of Property (b) — Suppression**

Suppose for contradiction that $\pi^*_{\text{DPC}}(k) \in R$ for some $k$ satisfying the hypothesis. Let $i^* = \pi^*_{\text{DPC}}(k) \in R$ and $i_{\text{alt}} = \arg\min_{i\notin R} C_{\text{base}}(i,k) \notin R$.

Construct $\tilde{\pi}$: $\tilde{\pi}(k) = i_{\text{alt}}$ and $\tilde{\pi}(k') = \pi^*_{\text{DPC}}(k')$ for $k' \ne k$. We show $\tilde{\pi}$ achieves strictly lower cost under $C_{\text{DPC}}$.

The cost difference at $k$:

$$\begin{aligned}\Delta &= C_{\text{DPC}}(i^*,k) - C_{\text{DPC}}(i_{\text{alt}},k)\\ &= [C_{\text{base}}(i^*,k) + \lambda_{\text{match}}\beta_{i^*}] - [C_{\text{base}}(i_{\text{alt}},k) + \lambda_{\text{match}}\beta_{i_{\text{alt}}}].\end{aligned}$$

Since $i^* \in R$, $\beta_{i^*} \ge \tau$; since $i_{\text{alt}} \notin R$, $\beta_{i_{\text{alt}}} \le \tau'$. Thus

$$\lambda_{\text{match}}(\beta_{i^*} - \beta_{i_{\text{alt}}}) \ge \lambda_{\text{match}}(\tau - \tau').$$

By definition of $i_{\text{alt}}$, $C_{\text{base}}(i_{\text{alt}},k) = \min_{i\notin R} C_{\text{base}}(i,k)$, and $C_{\text{base}}(i^*,k) \ge \min_{i\in R} C_{\text{base}}(i,k)$. Hence

$$C_{\text{base}}(i^*,k) - C_{\text{base}}(i_{\text{alt}},k) \ge -\gamma_k(R).$$

Combining: $\Delta \ge -\gamma_k(R) + \lambda_{\text{match}}(\tau - \tau')$.

By hypothesis $\lambda_{\text{match}}(\tau - \tau') > \gamma_k(R)$, so $\Delta > 0$. Thus $\tilde{\pi}$ has strictly lower total cost, contradicting optimality. Therefore $\pi^*_{\text{DPC}}(k) \notin R$. □

**Proof of Property (c) — Stability**

Property (c) is a sensitivity result for linear assignment. The optimal assignment is piecewise-constant in the cost matrix, with pieces separated by hyperplanes. The minimum gap of $C_{\text{base}}$ is

$$\gamma_{\min}(C_{\text{base}}) = \min_{\pi\ne\pi^*(C_{\text{base}})}\left[\sum_k C_{\text{base}}(\pi(k),k) - \sum_k C_{\text{base}}(\pi^*(k),k)\right].$$

For $C_{\text{DPC}} = C_{\text{base}} + \lambda_{\text{match}} 1_M^T \otimes \beta$, the total cost of any assignment $\pi$ under $C_{\text{DPC}}$ differs from its cost under $C_{\text{base}}$ by $\lambda_{\text{match}}\sum_k \beta_{\pi(k)}$.

For $\beta, \beta'$ with $\|\beta - \beta'\|_\infty \le \Delta\beta$, the change in total cost of any assignment is bounded by $\lambda_{\text{match}}\cdot M\cdot\Delta\beta$. If $\lambda_{\text{match}}\cdot M\cdot\Delta\beta < \gamma_{\min}$, the modulation cannot break the gap, so the optimal assignment under both modulated matrices coincides with $\pi^*(C_{\text{base}})$. Therefore $\pi^*_{\text{DPC}}(\beta) = \pi^*_{\text{DPC}}(\beta')$. □

**Sharpness.** The bound in (b) is sharp: when $\lambda_{\text{match}}(\tau - \tau') = \gamma_k(R)$ exactly, $\Delta = 0$ and either assignment is optimal. The bound in (c) is conservative; tighter bounds depending on the specific structure of $C_{\text{base}}$ are achievable but do not improve the qualitative conclusion.

## H. Proof of Proposition 3

We calculate $L_\tau$ explicitly for each task.

**Detection.** Injection: $s_i' = s_i - \lambda_{\text{obj}}\beta_i$ and $z_i' = z_i - \lambda_{\text{cls}}\beta_i 1$. The change in calibrated logits under perturbation $\Delta\beta_i$ is bounded by $\max(\lambda_{\text{obj}}, \lambda_{\text{cls}}) \cdot \Delta\beta_i$. Since $\beta_i$ is Lipschitz in $I$ (Proposition 2 with constant 1): $L_{\text{det}} \leq \max(\lambda_{\text{obj}}, \lambda_{\text{cls}})$.

**Segmentation.** Mask-pooled $\beta_i^{(\text{seg})} \to (1 - \gamma_{\text{seg}}\beta_i^{(\text{seg})})$. Mask pooling is non-expansive (Lipschitz 1). The multiplicative factor is Lipschitz with constant $\gamma_{\text{seg}}$: $L_{\text{seg}} \leq \gamma_{\text{seg}}$.

**Pose.** $\hat{\sigma}_j^{(\text{aug})} = \hat{\sigma}_j(1 + \lambda_{\text{pose}} I_{\text{img}})$. Pointwise sampling is non-expansive. The multiplicative factor is Lipschitz with constant $\hat{\sigma}_j\lambda_{\text{pose}}$. Bounding $\hat{\sigma}_j$ by $\sigma_{\max}$: $L_{\text{pose}} \leq \sigma_{\max}\lambda_{\text{pose}}$.

**OBB.** Angle-stability bounded by $|\Delta\theta_{\max}|$: $L_{\text{obb}} \leq |\Delta\theta_{\max}|$.

**Classification.** Global multiplicative attenuation $(1 - \lambda_{\text{cls-global}}\bar{\beta})$. Spatial average $\bar{\beta}$ is Lipschitz with constant 1; attenuation Lipschitz with $\lambda_{\text{cls-global}}$: $L_{\text{cls}} \leq \lambda_{\text{cls-global}}$.

**Summary.**

$$L_{\max} = \max\{\lambda_{\text{obj}}, \lambda_{\text{cls}}, \gamma_{\text{seg}}, \sigma_{\max}\lambda_{\text{pose}}, |\Delta\theta_{\max}|, \lambda_{\text{cls-global}}\}.$$

This is finite under standard hyperparameter ranges, establishing the proposition. □

## I. Proof of Theorem 5

By Theorem 1, for $x_0 \sim p_0$ the expected residual at $(u, v)$ is

$$\mathbb{E}[\|\Delta_k(u,v)\|^2 \mid x_0] = \sigma(t_k)^2\,\mathbb{E}[\|\nabla \log p_{t_k}\|^2 \mid x_0] + C_{u,v}.$$

For $x_0 \sim p_0$, $\|\nabla \log p_{t_k}\|$ is bounded almost surely (the natural-image distribution has bounded score on its support). By Theorem 2 and the $K$-probe averaging plus concentration of measure, the suspicion field $R_{\text{deploy}}$ at any location is bounded above by a constant $M_{\text{clean}}$ almost surely.

After per-image normalization, the field is rescaled to $[0, 1]$. Under $p_0$, the field's typical value is small. The probability that $\beta_i$ exceeds threshold $\tau > 0$ for natural inputs is bounded:

$$\mathbb{P}[\beta_i > \tau \mid x_0 \sim p_0] = O(\exp(-c\tau^2 K))$$

for some $c > 0$ depending on score-function moments. This probability $\to 0$ as $\tau \to 0$.

By Theorem 3(a), if $\beta = 0$ exactly, $\pi^*_{\text{DPC}} = \pi^*$. By Theorem 3(c), if $\|\beta\|_\infty < \gamma_{\min}(C_{\text{base}})/(\lambda_{\text{match}}M)$, then $\pi^*_{\text{DPC}} = \pi^*$.

Combining: for $x_0 \sim p_0$, with probability $\to 1$ as $\tau \to 0$, the suspicion vector $\beta$ satisfies $\|\beta\|_\infty < \gamma_{\min}/(\lambda_{\text{match}}M)$, and therefore $\pi^*_{\text{DPC}} = \pi^*$. The defense does not redirect the assignment on natural inputs in the limit. □

**Discussion.** Theorem 5 establishes the structural clean-accuracy preservation property of the framework. The conditions are stated explicitly: optimal denoiser (well-trained on natural-image distributions provides adequate approximation); calibration constants per Theorem 3 guidance; asymptotic in $\tau$. The guarantee is qualitatively different from PatchCleanser's certification: it bounds the probability of clean-input fidelity directly, not the worst case over patches.