When Graph Anomalies Learn to Hide: Test-Time Cloaking in Graph-Level Anomaly Detection

Jinyu Cai*
Institute of Data Science
National University of Singapore
jinyucai@nus.edu.sg

Glynnis Lim
Institute of Data Science
National University of Singapore
glynnis@nus.edu.sg

See-Kiong Ng
Institute of Data Science
National University of Singapore
seekiong@nus.edu.sg

Abstract

Graph-level anomaly detection (GLAD) is increasingly deployed in high-stakes settings, where its performance is evaluated on static, clean test graphs at deployment. This creates a critical gap in practice, as anomalous graph inputs may be structurally modified after deployment to evade a fixed detector. We formalize this challenge as *test-time graph cloaking*: an unsupervised score-minimization attack within a bounded graph-edit budget. We introduce score-directed graph cloaking (SGC), an architecture-agnostic white-box attack that operates solely on the scalar anomaly score, making it applicable across diverse GLAD architectures. To isolate adaptive evasion from generic perturbation sensitivity, we pair SGC with budget-matched random edits and identify a distinct directedness gap. While random structural noise largely preserves anomaly separation, score-directed edits can drive anomalous graphs into the normal-score region. To mitigate this, we propose adversarial score training (AST), a normal-only defense that smooths the score landscape via adversarial local score regularization on normal training graphs. Our experiments across diverse benchmarks and GLAD detectors demonstrate that AST empirically improves cloaking robustness against adaptive evasion without relying on anomaly labels.

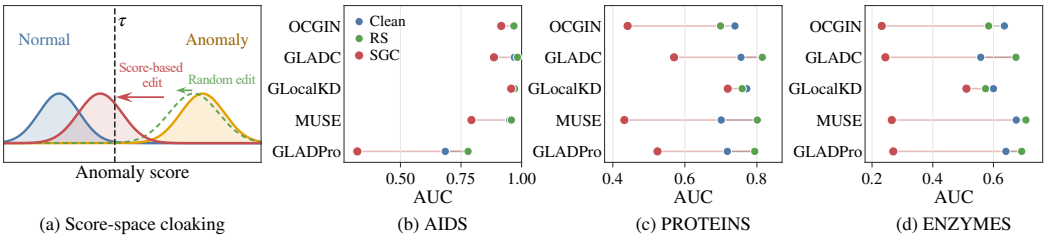
1 Introduction

Graph-level anomaly detection (GLAD) [1, 29, 46, 22, 34, 6] evaluates whether an entire graph (e.g., a molecule or a biological subnetwork) deviates from a learned distribution of normality. In the common unsupervised setting, a detector is trained on normal graphs and subsequently assigns an anomaly score to each test graph. Although recent GLAD research based on one-class objectives [50, 35, 48], knowledge distillation [28], contrastive learning [27, 26], and reconstruction-based scoring [23] have advanced clean-test performance, current benchmarks such as UB-GOLD focus on static, clean-test comparisons [37]. This protocol leaves a critical question unresolved: *how reliable are these detectors when anomalous inputs adapt after deployment?*

Such test-time adaptivity is a major concern in high-stakes deployment. For example, in molecular or biological graph screening pipelines [38, 21, 43], an anomalous graph input may be structurally modified before inspection to reduce the anomaly score. Prior works have shown that GNNs can be

*Corresponding Author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Score-space cloaking (b) AIDS (c) PROTEINS (d) ENZYMES

Figure 1: **Motivation for test-time graph cloaking.** (a) Clean anomaly scores may be separated from normal scores, but this separation need not be stable under score-directed structural edits. (b–d) Under the same 10% edge-flip budget, random-structure (RS) can preserve or even improve AUC relative to clean evaluation, whereas SGC causes substantial performance degradation.

sensitive to structural perturbations [10, 36, 49], suggesting that such edits may expose low-score directions in the detector. In such cases, the anomaly need not become normal in any semantic sense; it only needs to manipulate the scoring logic of the detector. We study this failure mode as *test-time graph cloaking*: after training is completed, an anomalous test graph modifies its structure within a small edge-edit budget to minimize the anomaly score.

Test-time cloaking is distinct from training contamination in GLAD, where anomalies leak into the normal training set [12, 33, 9]. It also differs from standard supervised graph attacks, which typically rely on labels, target classes, cross-entropy margins, or classifier-specific losses [55, 41, 4], none of which are available in unsupervised GLAD. Existing unsupervised graph attacks generally target only representations or node-level decisions [51, 17]. In unsupervised GLAD, there is a wide diversity of training objectives. However, they share a common interface, the scalar anomaly score, where higher values denote more "anomalous". We therefore ask whether the scalar score has exploitable local directions under small structural edits.

A motivating failure mode. Clean-test evaluations show that anomalies are separable from normal data at the fixed observed input. Test-time cloaking poses a different question: whether the structural evidence supporting that separation remains stable under adaptive edits. Figure 1 illustrates the gap: budget-matched random edits probe uninformed topology noise, whereas score-directed edits search for local structural changes that reduce the detector's scalar anomaly score. A detector can therefore appear stable under random perturbations while remaining vulnerable to adaptive cloaking. The vulnerability is not generic graph-noise sensitivity, but exploitable low-score directions under a small edge-edit budget. We call the resulting separation between budget-matched random edits and score-directed edits the *directedness gap*.

In this paper, we establish test-time cloaking as a robustness axis for unsupervised GLAD. We formalize cloaking as score minimization under a bounded edge-edit budget and introduce score-directed graph cloaking (SGC), a white-box attack that uses only the oriented anomaly score and therefore can be applied across diverse GLAD detectors. Given that unsupervised GLAD provides neither anomaly labels nor anomalous training graphs, we then ask what can be defended within the normal-only regime. We propose adversarial score training (AST), a detector-agnostic regularizer that adversarially smooths local score variations around normal training graphs. AST does not certify anomaly-side robustness as the attacker operates on graphs that the defender never observes, but instead tests if the normal-side smoothing can transfer partially through the learned graph representation. Empirically, AST improves robust AUC and reduces threshold evasion across detectors, while the remaining gap exposes the asymmetry of normal-only defense.

We summarize our contributions as follows:

- We formulate **test-time graph cloaking** as a robustness problem for unsupervised GLAD, distinguishing it from training contamination and existing graph adversarial attacks.
- We propose SGC, a detector-agnostic score-directed attack, and introduce the directedness gap to distinguish adaptive evasion from budget-matched random perturbation sensitivity.
- We introduce AST, a normal-only adversarial score-smoothing defense, and empirically show that clean evaluation substantially overstates evasion robustness, while AST partially closes the cloaking gap under the main SGC protocol.

2 Methodology

2.1 Problem Formulation and Threat Model

Notation. A graph is denoted $G = (V, E, X)$ with $n = |V|$ nodes and node feature matrix $X \in \mathbb{R}^{n \times d}$. Its adjacency matrix $A \in \{0,1\}^{n \times n}$ is symmetric and self-loop-free: $A = A^\top$, $A_{ii} = 0$. Let

$$\mathcal{U}_n = \{(i,j) : 1 \le i < j \le n\}, \qquad M_n = |\mathcal{U}_n| = n(n-1)/2$$

be the unordered candidate edge set. We write $s_\theta(A, X)$ for the score of $G(A, X)$, and use the same shorthand when the detector is evaluated on a relaxed weighted adjacency $\widetilde{A}$. All edit variables are defined on $\mathcal{U}_n$, and the full adjacency is always recovered by symmetrization with a zero diagonal.

Unsupervised GLAD. The training set $\mathcal{D}_{\text{tr}} = \{G_i\}_{i=1}^N$ contains only normal graphs. A GLAD model learns a scoring function

$$s_\theta : \mathcal{G} \to \mathbb{R}, \tag{1}$$

where larger values indicate stronger abnormality. Baselines may natively output reconstruction errors, one-class distances, likelihoods, distillation discrepancies, or prototype scores. When a baseline uses the opposite convention, we orient its output so that every detector shares the same larger-is-more-anomalous convention. This orientation is not cosmetic: SGC minimizes the score, AST regularizes score increases, and ranking metrics are computed from the same oriented scores. A threshold τ gives $\hat{y}(G;\tau) = \mathbf{1}\{s_\theta(G) > \tau\}$. In all experiments, τ is the 95th percentile of validation-normal scores, which is used only for threshold metrics such as ASR.

Test-time graph cloaking. After training, the detector is fixed. Given an anomalous test graph $G_a = (A, X)$, a cloaking adversary edits only the structure and keeps X unchanged [3, 16]. The goal is not to make the graph semantically normal. It is to move the detector's scalar score into the normal-score region while changing only a small number of edges. For an edit ratio $\rho \in [0,1]$, the budget is

$$k_E(G;\rho) = \begin{cases} \max\{1, \lfloor \rho |E| \rfloor\}, & \rho > 0 \text{ and } |E| > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The budget scales with the observed graph size rather than with the complete candidate set M_n, so ρ measures a relative structural change instead of a fraction of all possible unordered pairs. This avoids assigning sparse graphs unrealistically large insertion budgets. The floor with $\max\{1, \cdot\}$ avoids a degenerate zero budget for small graphs when $\rho > 0$. Let $b \in \{0,1\}^{M_n}$ be an upper-triangular flip mask. A single unit of budget either inserts a missing edge or deletes an existing edge through the symmetric XOR operation

$$(A \oplus b)_{ij} = (A \oplus b)_{ji} = A_{ij} \oplus b_{ij}, \quad (i,j) \in \mathcal{U}_n, \qquad (A \oplus b)_{ii} = 0. \tag{3}$$

Test-time cloaking is the score-minimization problem

$$\min_{b \in \{0,1\}^{M_n}} s_\theta(A \oplus b, X) \quad \text{s.t.} \quad \|b\|_0 \le k_E(G_a;\rho). \tag{4}$$

A threshold evasion occurs when an anomaly is detected before editing, $s_\theta(G_a) > \tau$, but is pushed below the threshold after editing, $s_\theta(A \oplus b, X) \le \tau$. Ranking-based degradation is evaluated on the full test set, while ASR is evaluated only over anomalies that were detected before the attack.

Threat model. The attack is per-graph, test-time, and white-box: the attacker can differentiate the deployed detector with respect to a relaxed adjacency, but uses only the scalar anomaly score as its objective. It does not use class labels, detector-specific losses, training-time poisoning, or backdoors. We focus on structure perturbations because they are the most universal edit type across GLAD datasets and avoid validity assumptions tied to specific feature semantics (e.g., chemical validity for molecular features). The main protocol uses a general structural edit space, complemented by edits on AIDS that satisfy maximum-valence constraints and preserve component counts in Section 3.3. Feature-space attacks require additional dataset-specific edit rules. The white-box setting probes exploitable directions in the detector's local score surface. Appendix D.7 also evaluates query-limited cloaking using scalar-score access.

2.2 Score-Directed Graph Cloaking

Solving Eq. (4) exactly requires searching $\sum_{j=0}^{k_E} \binom{M_n}{j}$ edit sets. Unlike supervised graph attacks, unsupervised GLAD provides no class label or classifier margin; the only uniform interface is the scalar anomaly score. SCORE-DIRECTED GRAPH CLOAKING (SGC) therefore approximates Eq. (4) by optimizing a continuous flip intensity $q \in [0,1]^{M_n}$ and then rounding the result back to a binary graph. For $(i,j) \in \mathcal{U}_n$, define

$$\widetilde{A}_{ij}(q) = A_{ij} + (1 - 2A_{ij})q_{ij}, \qquad \widetilde{A}_{ji}(q) = \widetilde{A}_{ij}(q), \qquad \widetilde{A}_{ii}(q) = 0. \tag{5}$$

At binary vertices, $\widetilde{A}(q) = A \oplus q$. The sign $(1 - 2A_{ij})$ is the key: increasing q_{ij} inserts a missing edge when $A_{ij} = 0$ and deletes an existing edge when $A_{ij} = 1$. Thus, additions and deletions are optimized in one coordinate system.

Budget-respecting relaxation. SGC searches over the capped simplex

$$\mathcal{C}_G^{\mathrm{atk}}(\rho) = \left\{ q \in [0,1]^{M_n} : \mathbf{1}^\top q \le k_E(G;\rho) \right\}. \tag{6}$$

This keeps the relaxed attack on the same total edit budget as the discrete attacker. A pure box relaxation could spread large intensity over many unrealizable edits, and the capped simplex prevents that mismatch. We project onto this set at every step, using the Euclidean projection detailed in Appendix B.2.

Relaxed score descent. Starting from $q^{(0)} = \mathbf{0}$, SGC performs projected gradient descent on the oriented anomaly score [41, 32]. Because score scales vary widely across GLAD objectives, we normalize the gradient by its mean absolute magnitude:

$$\widehat{g}^{(t)} = \frac{\nabla_q s_\theta(\widetilde{A}(q^{(t)}), X)}{M_n^{-1} \sum_{m=1}^{M_n} \left| \nabla_q s_\theta(\widetilde{A}(q^{(t)}), X)_m \right| + \varepsilon_g}, \qquad q^{(t+1)} = \Pi_{\mathcal{C}_{G_a}^{\mathrm{atk}}(\rho)} \left(q^{(t)} - \eta \widehat{g}^{(t)} \right). \tag{7}$$

All gradients are taken with respect to the oriented anomaly score, so descent always means moving toward lower abnormality. The fixed detector-agnostic attack configuration is reported in Appendix C.

Binary rounding. The relaxed solution is used only to rank candidate edits. Let

$$\mathcal{P} = \mathrm{TopK}\left(q^{(T)}, \min\{M_n, \lceil c\, k_E(G_a;\rho) \rceil\} \right) \tag{8}$$

be the candidate pool, where $c \ge 1$ is fixed across experiments. Starting from the clean adjacency, SGC repeatedly tests each unused candidate as a one-edge flip, commits the flip that yields the lowest binary-graph score, and stops when the budget is exhausted. If the rounded graph does not reduce the clean score, SGC returns the clean graph as a no-op. Hence, the realized attack always uses at most $k_E(G_a;\rho)$ flips. All reported attack metrics are evaluated on the final binary graph, never on $\widetilde{A}$. Appendix B.1 gives the full algorithm, the chain-rule gradient through $\widetilde{A}$, and a relaxed-to-discrete diagnostic.

Directedness gap. To distinguish adaptive evasion from ordinary perturbation sensitivity, we compare SGC with a budget-matched random-structure (RS) control. For each anomalous graph, RS samples $k_E(G_a;\rho)$ unordered pairs uniformly without replacement from $\mathcal{U}_n$ and flips them symmetrically, while normal test graphs are left unchanged under both protocols. The two evaluations share the same budget, edit space, and binary granularity, differing only in edit selection. For any metric $\mathcal{M}$ where larger is better, let $\mathcal{M}_{\mathrm{SGC}}(\rho)$ denote the SGC value and

$$\mathcal{M}_{\mathrm{rand}}(\rho) = \frac{1}{R} \sum_{r=1}^{R} \mathcal{M}_{\mathrm{rand}}^{(r)}(\rho) \tag{9}$$

be the seed-averaged RS value. We define

$$\mathrm{DG}_\rho(\mathcal{M}) = \mathcal{M}_{\mathrm{rand}}(\rho) - \mathcal{M}_{\mathrm{SGC}}(\rho). \tag{10}$$

For smaller-is-better metrics such as ASR, we reverse the sign. A positive directedness gap suggests that, under the matched-budget protocol, the degradation is attributable to score-aligned edit selection rather than to the edit budget alone.

2.3 Adversarial Score Training

Classical adversarial training solves a supervised saddle point such as $\min_\theta \mathbb{E}_{(x,y)}[\max_\delta \mathcal{L}(f_\theta(x+\delta), y)]$ [30, 14]. Normal-only GLAD breaks this template: there are no anomaly labels during training, and the cloaking adversary acts on anomalous graphs the defender never sees. ADVERSARIAL SCORE TRAINING (AST) therefore regularizes the one object available in training: local score increases around normal graphs.

Normal-side adversary. For a normal training graph $G = (A, X)$, define

$$\mathcal{C}_G(\rho_{\mathrm{tr}}) = \left\{ q \in [0,1]^{M_n} : \mathbf{1}^\top q \le k_E(G; \rho_{\mathrm{tr}}) \right\}. \tag{11}$$

The inner problem searches for a relaxed structure that makes this normal graph look more anomalous:

$$q_\theta^+(G) \approx \arg \max_{q \in \mathcal{C}_G(\rho_{\mathrm{tr}})} s_\theta(\widetilde{A}(q), X). \tag{12}$$

We solve it by projected ascent with best-iterate selection and initialize the best iterate at $q = \mathbf{0}$. Thus, if the inner solver finds no score-increasing edit, the adversarial example falls back to the clean graph and the penalty below is zero.

Relative score-smoothing penalty. Given $q_\theta^+(G)$, AST penalizes the upward score change relative to the clean score scale:

$$\mathcal{R}_{\mathrm{AST}}(\theta; G) = \left(\frac{s_\theta(\widetilde{A}(q_\theta^+(G)), X) - s_\theta(A, X)}{\mathrm{sg}(\max\{|s_\theta(A, X)|, \varepsilon\})} \right)^2, \tag{13}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient and $\varepsilon = 1$ avoids unstable amplification near zero. The relative normalization is important because raw anomaly scores from one-class, distillation, reconstruction, and prototype-based detectors can differ by orders of magnitude. Without it, the same λ would mean different regularization strengths for different baselines. The training objective is

$$\min_\theta \ \mathbb{E}_{G \sim \mathcal{D}_{\mathrm{tr}}} \left[\mathcal{L}_{\mathrm{base}}(\theta; G) + \lambda \mathcal{R}_{\mathrm{AST}}(\theta; G) \right], \tag{14}$$

where $\mathcal{L}_{\mathrm{base}}$ is the detector's native unsupervised loss. As in first-order adversarial training, the outer update treats $q_\theta^+(G)$ as fixed. Appendix B.2 gives the solver and projection details.

Why the smoothing is one-sided. AST does not inject random graph noise. The inner maximization only searches for edits that increase the anomaly score of a normal graph. Since $q = \mathbf{0}$ is feasible, best-iterate selection guarantees $s_\theta(\widetilde{A}(q_\theta^+), X) \ge s_\theta(A, X)$, with equality when no such direction is found. The squared penalty, therefore, suppresses upward score excursions on normal graphs, vanishes on already-flat neighborhoods to concentrate capacity on locally steep regions [31], and, after the common larger-is-more-anomalous orientation, reduces detector-specific score-scale effects.

Scope and falsifiability. AST directly controls only relaxed neighborhoods of normal training graphs. It does not certify robustness for unseen anomalous graphs. Its benefit can only arise indirectly if the encoder and scoring head propagate normal-side smoothness to nearby regions of representation space. For that reason, all AST-trained detectors are evaluated with the same external SGC attack used for vanilla detectors, with an evaluation-time attack stronger than the training inner loop (Appendix C). Appendix B.3 further specifies diagnostics for checking whether robustness gains reflect score smoothing rather than gradient masking.

3 Experiments

3.1 Experiment Configuration

Datasets. We evaluate on graph-level anomaly detection datasets from the TUDataset [2] under the UB-GOLD anomaly-detection protocol [37], including AIDS, PROTEINS, COX2, ENZYMES, and BZR. Dataset statistics and data split are reported in Appendix C.

[2] https://chrsmrrs.github.io/datasets/docs/home/

Table 1: Undefended GLAD models under random-structure (RS) and SGC perturbations. All results are five-fold mean ± standard deviation. Directedness gap (DG) is the performance gap between RS and SGC. Lower AUC and higher ASR indicate stronger attack effectiveness, while a larger DG indicates stronger score-directed vulnerability beyond random graph noise.

Dataset	Model	AUC↓			ASR ↑		DG ↑
		Clean	RS	SGC	RS	SGC	
AIDS	OCGIN	0.971±0.015	0.968±0.015	**0.916±0.032**	0.067±0.026	**0.294±0.050**	0.052
	GLADC	0.970±0.011	0.983±0.009	**0.886±0.024**	0.002±0.002	**0.448±0.084**	0.097
	GLocalKD	0.972±0.015	0.971±0.015	**0.957±0.019**	0.030±0.010	**0.104±0.044**	0.014
	MUSE	0.951±0.010	0.958±0.011	**0.793±0.021**	0.028±0.013	**0.835±0.079**	0.165
	GLADPro	0.685±0.039	0.779±0.025	**0.322±0.118**	0.227±0.228	**0.923±0.084**	0.457
PROTEINS	OCGIN	0.739±0.014	0.699±0.018	**0.441±0.022**	0.180±0.013	**0.680±0.069**	0.258
	GLADC	0.756±0.045	0.815±0.035	**0.570±0.058**	0.000±0.000	**0.748±0.104**	0.245
	GLocalKD	0.772±0.035	0.759±0.036	**0.719±0.043**	0.065±0.044	**0.186±0.057**	0.040
	MUSE	0.701±0.027	0.801±0.033	**0.432±0.070**	0.002±0.004	**0.953±0.064**	0.369
	GLADPro	0.718±0.051	0.794±0.040	**0.524±0.087**	0.045±0.022	**0.715±0.162**	0.270
ENZYMES	OCGIN	0.636±0.111	0.584±0.117	**0.232±0.068**	0.183±0.239	**0.617±0.139**	0.352
	GLADC	0.558±0.083	0.674±0.072	**0.244±0.103**	0.000±0.000	**1.000±0.000**	0.430
	GLocalKD	0.600±0.110	0.574±0.114	**0.511±0.116**	0.110±0.152	**0.250±0.250**	0.063
	MUSE	0.675±0.050	0.707±0.063	**0.265±0.113**	0.083±0.167	**0.750±0.200**	0.442
	GLADPro	0.641±0.052	0.693±0.062	**0.270±0.105**	0.050±0.071	**0.583±0.118**	0.423

Experimental settings. We evaluate each detector over five stratified folds under three test-time conditions: **Clean**, **RS** (random-structure perturbation), and **SGC** (score-directed graph cloaking). In attacked evaluations, only anomalous test graphs are perturbed, while node features, labels, training data, and trained model parameters remain fixed. Unless otherwise stated, RS and SGC use the same edge-flip budget $\rho = 0.10$, enabling a direct comparison between uninformed graph edits and targeted score minimization. Full split construction, threshold calibration, random-repeat protocol, and attack/defense hyperparameters are given in Appendix C.

Baselines and evaluation metrics. We include five graph-level anomaly detection baselines that cover complementary modeling paradigms, including OCGIN [50], GLADC [27], GLocalKD [28], MUSE [23], and GLADPro [42]. We implement each baseline method using the UB-GOLD [37] benchmark to ensure reproducibility and fairness.

We use AUROC, abbreviated as AUC, as the primary evaluation metric. AUPRC results are also reported in Appendix D.3. Additionally, we report attack success rate (ASR) [8] [3] as supplementary threshold evidence, which is computed among anomalous test graphs that are detected before attack under the validation-normal 95th-percentile threshold. If the denominator is zero, ASR is marked as N/A. To separate score-directed evasion from random perturbation effects, we define the directedness gap (DG) as $\mathrm{DG} = \mathrm{AUC}_{\mathrm{RS}} - \mathrm{AUC}_{\mathrm{SGC}}$. A larger positive value means that the score-directed attack is stronger than a random perturbation under the same budget. Ranking metrics are computed on the full fold test set; ASR is computed over the detected-anomaly subset as above.

3.2 Experimental Results under Test-Time Cloaking

Tables 1 and 6 (in Appendix D.1) evaluate undefended GLAD models under clean testing, random structure perturbation, and the proposed SGC attack. Our main finding is that clean test metrics are a weak proxy for test-time evasion robustness. For example, on the AIDS dataset, OCGIN, GLADC, and GLocalKD all achieve clean AUCs around 0.97, yet their SGC ASR ranges from 0.104 to 0.448. MUSE and GLADPro are more vulnerable under the same 10% edge-flip budget. Therefore, clean separability measures whether the observed anomaly is detectable, not whether the evidence supporting that decision is stable under score-directed edits.

The comparison with random-structure perturbation is crucial for interpreting this gap. Random structure perturbation often preserves AUC and sometimes improves it, whereas SGC is consistently worse than the budget-matched random baseline. This separates adaptive cloaking from generic graph-

[3] $\mathrm{ASR} = |\{a : s_\theta(G_a) > \tau \ \wedge \ s_\theta(G_a^\star) \le \tau\}| / |\{a : s_\theta(G_a) > \tau\}|$.

noise sensitivity: Random edits may introduce additional irregularity that makes anomalous graphs easier to sort, while SGC selects edges aligned with low-score directions of the deployed detector. The directedness gap further shows that vulnerability is architecture-dependent. GLocalKD has the smallest DG among all datasets, suggesting a lower locally exploitable surface in this evaluation setting, whereas GLADC, MUSE, and GLADPro can exhibit much larger gaps. The COX2/BZR rows in Table 6 show the same mechanism in a more extreme form: random edits can substantially improve AUC while score-directed edits nearly collapse several detectors. These observations indicate that robustness evaluation for GLAD should not stop at clean performance: a detector can rank clean anomalies well and be insensitive to random perturbations, but remains vulnerable to adaptive cloaking.

We further compare SGC with a query-limited attack and a one-step score-saliency attack on AIDS with OCGIN. Both achieve higher mean ASR and lower mean post-attack AUC than RS, with iterative SGC giving the strongest mean attack performance. This comparison shows that the vulnerability extends to several forms of score guidance, including direct score queries. Appendix D.7 gives the configurations and results.

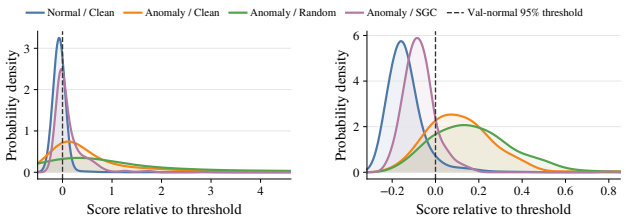
3.3 Validity-Constrained Cloaking and Independent Audits

To examine whether cloaking persists under additional structural restrictions, we evaluate OCGIN on AIDS with edits that respect atomic maximum valence, preserve the original number of connected components, and insert only single bonds. Constrained SGC and RS share these rules and the edge-edit budget $\rho = 0.10$. We then audit successful evasions with an independently trained GIN classifier f_{ext} that predicts the original dataset classes, together with the independent detectors GLADC and GLocalKD. These models are used only after editing. A jointly audited evasion retains the anomalous class under f_{ext} and remains flagged by both detectors. Appendix D.6 provides the audit details and a representative case.

Table 2 shows that SGC produces 48 evasions compared with 17 for RS under the same restrictions. The AUC directedness gap is 0.044, retaining approximately 85% of the unconstrained gap of 0.052. The constraints reduce SGC's ASR from 0.294 to 0.136, indicating that the admissible edit space affects attack strength. Among the successful evasions, 40 SGC outputs pass all three audits, compared with 13 for RS. The persistence of this difference under both structural restrictions and independent checks supports a benchmark-level evasion interpretation: the target detector can assign a low score to an edited graph whose anomalous designation is retained by the auditing models. Functional and biochemical equivalence require further domain-specific validation.

3.4 Score-Level Evidence of Graph Cloaking

Aggregate ranking metrics conceal where evasion occurs in score space. Figure 2 resolves this by plotting per-graph score densities on AIDS for two representative detectors, re-centered at the validation-normal 95th-percentile threshold so that $s(G) - \tau < 0$ marks the normal-score region.



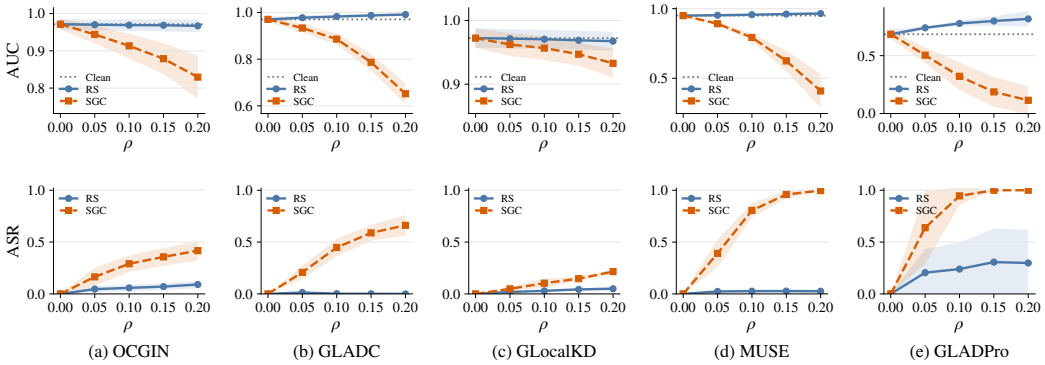
(a) GLADC (b) MUSE

Figure 2: Score distributions under clean, random-structure perturbation, and SGC evaluations. The x-axis reports detector scores relative to the validation-normal 95th percentile threshold, i.e., $s(G) - \tau$, which is denoted by the dashed vertical line at zero. The y-axis shows kernel density estimates, normalized so that each distribution integrates to one. Lower values indicate scores closer to the normal-score region.

Three patterns are visually consistent and align with the directedness gap reported in Table 1. **(i)** Clean anomaly scores concentrate to the right of the threshold, so passive separation is not the bottleneck. **(ii)** Random structural edits leave the anomaly density largely unchanged: uninformed perturbations tend to preserve or even amplify the anomalous structural signal. **(iii)** SGC produces a qualitatively different effect: the

Table 2: Constrained attacks and independent audits on AIDS with OCGIN at $\rho = 0.10$. AUC and ASR are five-fold means $\pm$ standard deviations. Evasion counts are pooled over 354 clean-detected anomalies. Jointly audited counts include successful evasions that pass all three independent checks.

Attack	AUC ↓	ASR ↑	Evasions	Jointly audited
Constrained RS	0.969 ± 0.015	0.048 ± 0.041	$17/354$	$13/354$
Constrained SGC	0.925 ± 0.022	0.136 ± 0.038	$48/354$	$40/354$



(a) OCGIN (b) GLADC (c) GLocalKD (d) MUSE (e) GLADPro

Figure 3: Budget sensitivity of RS and SGC for different GLAD models. The top row reports AUC and the bottom row reports ASR as the edge-flip budget ρ increases from 0 to 0.20. Note that the gray dotted lines in the top row denote clean AUC without perturbation.

anomaly mode shifts leftward into the sub-threshold region without comparable broadening, so the distribution is relocated rather than dispersed.

Generally, a purely noise-driven failure would more naturally appear as variance inflation, whereas the plotted failures in Figure 2 instead show coordinated score shifts. This indicates that small edit budgets can find coherent score-decreasing directions. Thus, we can draw two consequences. First, two detectors with near-identical clean-detection capabilities may exhibit different local score topologies, so clean rank quality alone does not determine the margin against adversarial edits. Second, because the displacement is direction-aligned rather than diffuse, the defense signal should directly regularize score changes along adversarial structural directions, motivating AST in Section 2.3.

3.5 Sensitivity to the Edit Budget

Table 1 fixes $\rho = 0.10$, which leaves two questions open: **(i)** is the directedness gap an artifact of this particular budget, and **(ii)** how does evasion scale with adversary capability? To answer these questions, we test all five GLAD methods with $\rho \in \{0, 0.05, 0.10, 0.20\}$ on AIDS.

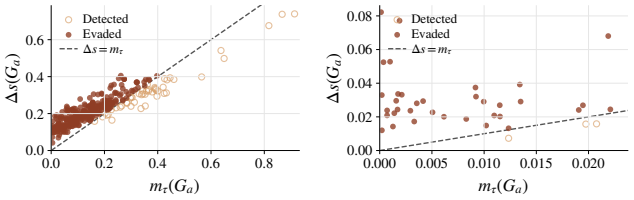
The positive budgets $\rho \in \{0.05, 0.10, 0.20\}$ correspond to mean realized edit counts of 1.63, 3.61, and 7.65 per attacked anomaly on AIDS, respectively, including successful and unsuccessful attacks. Thus, the budget sweep measures the effect of a small number of discrete structural changes. Appendix C reports the edit counts for all five datasets.

Figure 3 shows the results. Across all detectors, the SGC curves degrade as the budget increases, while the RS curves stay close to their clean baseline. In several cases, the RS curves (e.g., GLADC and MUSE) even drift upward with ρ. The two curves therefore diverge as the budget grows, ruling out the alternative that the reported gaps are an edge case of $\rho = 0.10$. Crucially, even at $\rho = 0.05$, the gap is already pronounced for methods such as MUSE and GLADPro. This indicates that score-directed edits do not require budgets near graph-rewriting to be effective. Additionally, we observe that the curves of GLocalKD remain nearly flat across the full ρ sweep, possibly due to the implicit smoothing induced by knowledge distillation. OCGIN and GLADC degrade approximately linearly, which indicates a roughly constant marginal score drop per edit. MUSE and GLADPro saturate by $\rho \approx 0.10$, suggesting that most attainable evasion concentrates in a small set of high-leverage edges.

3.6 Margin-Decrease Analysis of Threshold Evasion

ASR quantifies whether anomalies evade detection, but not why: was the clean anomaly signal already weak, or did the attack induce a large score drop? For a anomalous graph G_a detected under the clean setting, let $m_\tau(G_a) = s_\theta(G_a) - \tau$ denote its clean anomaly margin and $\Delta_s(G_a) = s_\theta(G_a) - s_\theta(G_a^{\mathrm{SGC}})$ denote its SGC-induced score decrease. By construction, threshold evasion occurs iff $\Delta_s(G_a) \geq m_\tau(G_a)$. Figure 4 plots both quantities for two representative baselines on AIDS, with the diagonal $\Delta_s = m_\tau$ separating evaded from still-detected graphs.

The two baselines fail with distinct signatures. MUSE spans a wide range of clean margins ($m_\tau \in [0, 0.8]$), and Δ_s scales approximately with m_τ, leaving many points just above the diagonal. This suggests that SGC exposes a margin-proportional descent direction: it can cancel the available clean margin even for confidently detected anomalies. GLADPro shows the opposite pattern. Its clean margins concentrate near zero ($m_\tau \lesssim 0.025$), while the induced decreases are substantially larger and only weakly correlated with m_τ. Here, near-universal evasion reflects borderline clean detection rather than unusually large attack-induced drops.



(a) MUSE (b) GLADPro

Figure 4: Margin-decrease decomposition of threshold evasion on AIDS. Each point is a clean-detected anomalous graph attacked by SGC at $\rho = 0.10$. The axes report the clean anomaly margin $m_\tau(G_a) = s_\theta(G_a) - \tau$ and the SGC-induced score decrease $\Delta s(G_a) = s_\theta(G_a) - s_\theta(G_a^{\mathrm{SGC}})$. The dashed diagonal is the evasion boundary $\Delta s = m_\tau$. The filled points above this line are successfully cloaked, while hollow points remain detected after SGC.

This decomposition exposes a blind spot in reporting only AUC or ASR. Comparable evasion rates can arise from qualitatively different mechanisms: local score sensitivity to structural edits, or anomaly scores separated from the threshold by only narrow margins. These mechanisms call for different remedies: the former requires smoothing the response of local scores to structural edits, while the latter requires amplifying the underlying anomaly signal.

3.7 Defense Results of AST

In this section, we evaluate whether AST improves robustness under the same SGC protocol as Table 1. Tables 3 and 7 (in Appendix D.1) show the experimental results, where a consistent trend is observed across all 25 dataset–model pairs: AST increases SGC AUC and reduces SGC ASR in every case, indicating that normal-only score smoothing provides a useful defense signal without anomaly labels. The improvements are substantial for detectors with highly steerable score surfaces. For example, the SGC AUC of GLADPro on COX2 improves from 0.023 to 0.346, GLADC on BZR from 0.136 to 0.417, and MUSE on BZR from 0.326 to 0.552. On easier settings such as AIDS, AST nearly eliminates threshold evasion for OCGIN and GLocalKD, reducing ASR to 0.003 and 0.020, respectively. Additionally, the overall improvement in GLocalKD is the most modest, consistent with its smaller directedness gap in Table 1. This suggests that when the vanilla score surface is already less locally exploitable, there is less attack-induced degradation for smoothing to recover. Yet AST does not fully close the cloaking gap: several difficult pairs still exhibit low AUC or high ASR after defense. This remaining vulnerability is expected, because AST regularizes neighborhoods of normal training graphs, whereas SGC attacks anomalous graphs unseen during training. Thus, AST should be viewed as an effective detector-agnostic baseline for normal-only cloaking defense, rather than a complete solution to test-time cloaking.

Additional experiments connect these gains to representation sensitivity and threshold behavior. AST reduces OCGIN's scale-normalized encoder sensitivity on anomalous test graphs by approximately 75% on AIDS and 34% on ENZYMES. On AIDS, OCGIN and GLADC maintain normal-test FPR close to 5% while improving clean-anomaly TPR. An adaptive attack that selects between score minimization and a normal-centroid objective yields $2/397$ evasions for AST-trained OCGIN, with post-attack AUC 0.996 ± 0.006. Appendix D.8 presents these complementary checks.

Table 3: Effect of adversarial score training (AST) under SGC. All results are reported as five-fold mean ± standard deviation. Δ SGC AUC and Δ SGC ASR denote AST minus Vanilla for the same dataset–model pair. Positive Δ SGC AUC and negative Δ SGC ASR indicate improved robustness.

Dataset	Model	Training	Clean AUC ↑	SGC AUC ↑	SGC ASR ↓	Δ SGC AUC ↑	Δ SGC ASR ↓
AIDS	OCGIN	Vanilla	0.971±0.015	0.916±0.032	0.294±0.050	+0.081	-0.291
		AST	**0.998±0.004**	**0.997±0.004**	**0.003±0.006**		
	GLADC	Vanilla	0.970±0.011	0.886±0.024	0.448±0.084	+0.062	-0.204
		AST	**0.989±0.005**	**0.948±0.016**	**0.244±0.126**		
	GLocalKD	Vanilla	0.972±0.015	0.957±0.019	0.104±0.044	+0.030	-0.084
		AST	**0.992±0.006**	**0.987±0.009**	**0.020±0.011**		
	MUSE	Vanilla	**0.951±0.010**	0.793±0.021	0.835±0.079	+0.049	-0.211
		AST	0.942±0.006	**0.842±0.059**	**0.624±0.063**		
	GLADPro	Vanilla	0.685±0.039	0.322±0.118	0.923±0.084	+0.140	-0.199
		AST	**0.789±0.025**	**0.462±0.083**	**0.724±0.107**		
PROTEINS	OCGIN	Vanilla	0.739±0.014	0.441±0.022	0.680±0.069	+0.235	-0.320
		AST	**0.772±0.017**	**0.676±0.047**	**0.360±0.096**		
	GLADC	Vanilla	0.756±0.045	0.570±0.058	0.748±0.104	+0.043	-0.110
		AST	**0.784±0.024**	**0.613±0.036**	**0.638±0.163**		
	GLocalKD	Vanilla	0.772±0.035	0.719±0.043	0.186±0.057	+0.023	-0.054
		AST	**0.789±0.030**	**0.742±0.041**	**0.132±0.050**		
	MUSE	Vanilla	0.701±0.027	0.432±0.070	0.953±0.064	+0.070	-0.142
		AST	**0.710±0.032**	**0.502±0.055**	**0.811±0.052**		
	GLADPro	Vanilla	0.718±0.051	0.524±0.087	0.715±0.162	+0.088	-0.239
		AST	**0.727±0.081**	**0.612±0.067**	**0.476±0.185**		
ENZYMES	OCGIN	Vanilla	**0.636±0.111**	0.232±0.068	0.617±0.139	+0.151	-0.054
		AST	0.620±0.124	**0.383±0.087**	**0.563±0.081**		
	GLADC	Vanilla	0.558±0.083	0.244±0.103	1.000±0.000	+0.120	-0.500
		AST	**0.633±0.087**	**0.364±0.072**	**0.500±0.107**		
	GLocalKD	Vanilla	0.600±0.110	0.511±0.116	0.250±0.250	+0.012	-0.033
		AST	**0.604±0.111**	**0.523±0.117**	**0.217±0.161**		
	MUSE	Vanilla	0.675±0.050	0.265±0.113	0.750±0.200	+0.160	-0.410
		AST	**0.704±0.090**	**0.425±0.258**	**0.340±0.204**		
	GLADPro	Vanilla	0.641±0.052	0.270±0.105	0.583±0.118	+0.081	-0.165
		AST	**0.649±0.034**	**0.351±0.073**	**0.418±0.162**		

4 Conclusion

In this paper, we study test-time graph cloaking, an overlooked robustness problem in unsupervised graph-level anomaly detection. We formulate cloaking as score minimization under a bounded graph-edit budget and propose SGC, a detector-agnostic white-box attack that uses only the oriented anomaly score. Our experiments show that clean-test performance can substantially overstate deployment robustness: small score-directed structural edits can move anomalous graphs toward the normal-score region, unlike budget-matched random edits. We further introduce AST, a normal-only score-smoothing defense that improves robust performance and reduces ASR under the main SGC protocol. The AIDS study further supports cloaking under structural constraints and independent anomaly-retention audits. AST remains an empirical mitigation, and extending the evaluation to broader domain constraints and adaptive attacks is an important direction for future work.

Acknowledgments and Disclosure of Funding

This research is supported by the National University of Singapore, Institute of Data Science. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the National University of Singapore.

References

[1] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the International Conference on Machine Learning*, pages 274–283. PMLR, 2018.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

[4] Aleksandar Bojchevski and Stephan Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *Proceedings of the International Conference on Machine Learning*, pages 695–704. PMLR, 2019.

[5] Jinyu Cai, Yunhe Zhang, and Jicong Fan. Self-discriminative modeling for anomalous graph detection. In *Proceedings of the International Conference on Machine Learning*, 2025.

[6] Jinyu Cai, Yunhe Zhang, Fusheng Liu, and See-Kiong Ng. Leveraging diffusion model as pseudo-anomalous graph generator for graph-level anomaly detection. In *Proceedings of the International Conference on Machine Learning*, 2025.

[7] Jinyu Cai, Yunhe Zhang, Pengyang Wang, and See-Kiong Ng. Moegad: A mixture-of-experts framework with pseudo-anomaly generation for graph-level anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(4):4473–4489, 2026.

[8] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. Ieee, 2017.

[9] Qingfeng Chen, Haojin Zeng, Jingyi Jie, Shichao Zhang, and Debo Cheng. Denoise: Learning robust graph representations for unsupervised graph-level anomaly detection. *arXiv preprint arXiv:2511.04086*, 2025.

[10] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *Proceedings of the International Conference on Machine Learning*, pages 1115–1124. PMLR, 2018.

[11] George Dasoulas, Kevin Scaman, and Aladin Virmaux. Lipschitz normalization for self-attention layers with application to graph neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 2456–2466. PMLR, 2021.

[12] Kaize Ding, Qinghai Zhou, Hanghang Tong, and Huan Liu. Few-shot network anomaly detection via cross-network meta-learning. In *Proceedings of the Web Conference*, pages 2448–2456, 2021.

[13] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the International Conference on Machine Learning*, pages 272–279, 2008.

[14] Fuli Feng, Xiangnan He, Jie Tang, and Tat-Seng Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2493–2504, 2019.

[15] Fernando Gama, Joan Bruna, and Alejandro Ribeiro. Stability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 68:5680–5695, 2020.

[16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[17] Filippo Guerranti, Zinuo Yi, Anna Starovoit, Rafiq Mazen Kamel, Simon Geisler, and Stephan Günnemann. On the adversarial robustness of graph contrastive learning methods. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.

[18] Yue Hou, Yingke Su, Junran Wu, and Ke Xu. Test-time graph ood detection via dynamic dictionary expansion and ood score calibration. In *Proceedings of the ACM International Conference on Multimedia*, pages 8845–8853, 2025.

[19] Yue Hou, He Zhu, Ruomei Liu, Yingke Su, Junran Wu, and Ke Xu. Redundancy-aware test-time graph out-of-distribution detection. *Advances in Neural Information Processing Systems*, 38:119899–119942, 2026.

[20] Yue Hou, He Zhu, Ruomei Liu, Yingke Su, Jinxiang Xia, Junran Wu, and Ke Xu. Structural entropy guided unsupervised graph out-of-distribution detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17258–17266, 2025.

[21] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33:22118–22133, 2020.

[22] Wei Ju, Siyu Yi, Yifan Wang, Zhiping Xiao, Zhengyang Mao, Hourun Li, Yiyang Gu, Yifang Qin, Nan Yin, Senzhang Wang, et al. A survey of graph neural networks in real world: Imbalance, noise, privacy and ood challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 3036–3055, 2025.

[23] Sunwoo Kim, Soo Yong Lee, Fanchen Bu, Shinhwan Kang, Kyungho Kim, Jaemin Yoo, and Kijung Shin. Rethinking reconstruction-based graph-level anomaly detection: limitations and a simple remedy. *Advances in Neural Information Processing Systems*, 37:95931–95962, 2024.

[24] Zhiqiang Kou, Junyang Chen, Xin-Qiang Cai, Xiaobo Xia, Ming-Kun Xie, Dong-Dong Wu, Biao Liu, Yuheng Jia, Xin Geng, Masashi Sugiyama, and Tat-Seng Chua. Positive–unlabeled reinforcement learning distillation for on-premise small models. In *Proceedings of the International Conference on Machine Learning*, volume 306, pages 60447–60479, 2026.

[25] Zhiqiang Kou, Haoyuan Xuan, Jingyu Zhu, Hailin Wang, Ming-kun Xie, Changwei Wang, Jing Wang, Yuheng Jia, and Xin Geng. Tail-aware reconstruction of incomplete label distributions with low-rank and sparse modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.

[26] Yixin Liu, Kaize Ding, Qinghua Lu, Fuyi Li, Leo Yu Zhang, and Shirui Pan. Towards self-interpretable graph-level anomaly detection. *Advances in Neural Information Processing Systems*, 36:8975–8987, 2023.

[27] Xuexiong Luo, Jia Wu, Jian Yang, Shan Xue, Hao Peng, Chuan Zhou, Hongyang Chen, Zhao Li, and Quan Z Sheng. Deep graph level anomaly detection with contrastive learning. *Scientific Reports*, 12(1):19867, 2022.

[28] Rongrong Ma, Guansong Pang, Ling Chen, and Anton van den Hengel. Deep graph-level anomaly detection by glocal knowledge distillation. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 704–714, 2022.

[29] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12012–12038, 2021.

[30] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the International Conference on Learning Representations*, 2018.

[31] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2018.

[32] Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are defenses for graph neural networks robust? *Advances in Neural Information Processing Systems*, 35:8954–8968, 2022.

[33] Chaoxi Niu, Guansong Pang, and Ling Chen. Graph-level anomaly detection via hierarchical memory networks. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 201–218. Springer, 2023.

[34] Hezhe Qiao, Hanghang Tong, Bo An, Irwin King, Charu Aggarwal, and Guansong Pang. Deep graph anomaly detection: A survey and new perspectives. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[35] Chen Qiu, Marius Kloft, Stephan Mandt, and Maja Rudolph. Raising the bar in graph-level anomaly detection. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2196–2203, 2022.

[36] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the Web Conference*, pages 673–683, 2020.

[37] Yili Wang, Yixin Liu, Xu Shen, Chenyu Li, Rui Miao, Kaize Ding, Ying Wang, Shirui Pan, and Xin Wang. Unifying unsupervised graph-level anomaly detection and out-of-distribution detection: A benchmark. In *Proceedings of the International Conference on Learning Representations*, 2025.

[38] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

[39] Zhihao Wu, Jinyu Cai, Yunhe Zhang, Jielong Lu, Zhaoliang Chen, Shuman Zhuang, and Haishuai Wang. Where graph meets heterogeneity: multi-view collaborative graph experts. *Advances in Neural Information Processing Systems*, 38:80096–80117, 2026.

[40] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

[41] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3961–3967, 2019.

[42] Zhenyu Yang, Ge Zhang, Jia Wu, Jian Yang, Shan Xue, Amin Beheshti, Hao Peng, and Quan Z Sheng. Global interpretable graph-level anomaly detection via prototype. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3586–3597, 2025.

[43] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

[44] Caiyang Yu, Wei Ju, Haixin Wang, Yifan Wang, and Ziyue Qiao. Dynamic multi-sample mixup with gradient exploration for open-set graph anomaly detection. In *Proceedings of the International Conference on Learning Representations*, 2026.

[45] Caiyang Yu, Wei Ju, Wenrui Wu, Yifan Wang, Ziyue Qiao, Chong Chen, Xian-Sheng Hua, and Xiao Luo. Fond: Fine-grained boundary mining with dual bridging for fair graph anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, pages 6630–6643, 2026.

[46] Ge Zhang, Zhenyu Yang, Jia Wu, Jian Yang, Shan Xue, Hao Peng, Jianlin Su, Chuan Zhou, Quan Z Sheng, Leman Akoglu, et al. Dual-discriminative graph neural network for imbalanced graph-level anomaly detection. *Advances in Neural Information Processing Systems*, 35:24144–24157, 2022.

[47] Yunhe Zhang, Jinyu Cai, Qi Hao, Pengyang Wang, and See-Kiong Ng. Escaping the homophily trap: A threshold-free graph outlier detection framework via clustering-guided edge reweighting. In *Proceedings of the International Conference on Learning Representations*, 2026.

[48] Yunhe Zhang, Yan Sun, Jinyu Cai, and Jicong Fan. Deep orthogonal hypersphere compression for anomaly detection. In *Proceedings of the International Conference on Learning Representations*, 2024.

[49] Kai Zhao, Qiyu Kang, Yang Song, Rui She, Sijie Wang, and Wee Peng Tay. Adversarial robustness in graph neural networks: A hamiltonian approach. *Advances in Neural Information Processing Systems*, 36:3338–3361, 2023.

[50] Lingxiao Zhao and Leman Akoglu. On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights. *Big Data*, 11(3):151–180, 2023.

[51] Yulin Zhu, Yuni Lai, Kaifa Zhao, Xiapu Luo, Mingquan Yuan, Jian Ren, and Kai Zhou. Binarizedattack: Structural poisoning attacks to graph-based anomaly detection. In *Proceedings of the IEEE International Conference on Data Engineering*, pages 14–26. IEEE, 2022.

[52] Jiaming Zhuo, Can Cui, Kun Fu, Bingxin Niu, Dongxiao He, Chuan Wang, Yuanfang Guo, Zhen Wang, Xiaochun Cao, and Liang Yang. Graph contrastive learning reimagined: Exploring universality. In *Proceedings of the ACM Web Conference*, pages 641–651, 2024.

[53] Jiaming Zhuo, Yuwei Liu, Yintong Lu, Ziyi Ma, Kun Fu, Chuan Wang, Yuanfang Guo, Zhen Wang, Xiaochun Cao, and Liang Yang. Dualformer: Dual graph transformer. In *Proceedings of the International Conference on Learning Representations*, 2025.

[54] Jiaming Zhuo, Yintong Lu, Hui Ning, Kun Fu, Bingxin Niu, Dongxiao He, Chuan Wang, Yuanfang Guo, Zhen Wang, Xiaochun Cao, and Liang Yang. Unified graph augmentations for generalized contrastive learning on graphs. In *Advances in Neural Information Processing Systems*, 2024.

[55] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018.

A Related Work

Graph-level anomaly detection. Graph-level anomaly detection (GLAD) [1, 29, 5, 7] identifies entire graphs that diverge from a learned normal distribution, as surveyed in the broader graph anomaly detection literature [1, 29, 34, 44, 45]. Recent unsupervised approaches advance clean-test performance through one-class and hypersphere objectives [50, 35, 48, 47], knowledge distillation [28, 24], contrastive learning [27, 26, 52, 54], reconstruction-based scoring [23, 25], and prototype-based interpretation [42], with benchmarks such as UB-GOLD standardizing evaluation across unsupervised anomaly and out-of-distribution settings [37, 20, 18, 19]. These efforts share a common assumption: anomalous graphs are evaluated as observed, without asking whether their separability persists once the detector is deployed. We target this complementary question by studying whether a trained detector remains reliable when anomalous test graphs are adaptively modified at inference. This regime is orthogonal to training-set contamination and robust representation learning for GLAD [12, 33, 9], which intervene at training time. Test-time graph cloaking instead holds the detector fixed and perturbs only anomalous inputs.

Adversarial robustness on graphs. Adversarial attacks on graph neural networks [40, 53, 39] have largely been developed in supervised or semi-supervised regimes, where structural perturbations target node labels, graph labels, embeddings, or classifier margins [10, 55, 41, 36, 49]. These methods typically rely on label information, target classes, surrogate classifiers, or cross-entropy objectives—signals that are unavailable in unsupervised GLAD. Existing unsupervised attacks likewise concentrate on node-level detection, embedding spaces, poisoning, or contrastive learning rather than on scalar graph-level scores [4, 51, 17]. Test-time graph cloaking departs from both lines along the interface and objective: rather than steering predictions toward a target class, the adversary applies bounded structural edits to reduce the detector's oriented anomaly score. Our SGC operationalizes this score-minimization view using only the scalar score shared across heterogeneous GLAD detectors, yielding a detector-agnostic attack. To separate adaptive evasion from generic perturbation sensitivity, we further introduce the directedness gap, which contrasts SGC against budget-matched random edits under an identical perturbation budget.

Adversarial training and smoothness regularization. Classical adversarial training defends classifiers by optimizing against worst-case perturbations on labeled inputs [30], and analogous graph defenses regularize GNNs against structural perturbations through adversarial training, Lipschitz control, and stability analysis [14, 15, 11, 32]. Virtual adversarial training extends this idea by enforcing local smoothness without requiring labels for the perturbation direction [31]. Unsupervised GLAD, however, imposes an asymmetric defense regime: only normal graphs are visible during training, whereas the cloaking adversary modifies anomalous graphs that the defender never observes. Our AST is designed for this normal-only setting—rather than enforcing supervised margins or training on attacked anomalies, it adversarially smooths upward score variations around normal graphs and asks whether this one-sided regularization transfers to unseen anomalies under an external SGC attack. AST should therefore be read as a detector-agnostic empirical baseline for normal-only cloaking defense, rather than as a certified guarantee against arbitrary adaptive adversaries.

B Additional Methodology Details

B.1 Properties and Implementation of SGC

Detector wrapper for weighted adjacencies. SGC needs gradients through a dense weighted adjacency $\widetilde{A} \in [0,1]^{n\times n}$. For each detector, we use a forward wrapper that accepts $\widetilde{A}$ without internal thresholding or silent binarization. We verified the wrapper by checking that $s_\theta(A, X)$ and $s_\theta(\widetilde{A}(\mathbf{0}), X)$ agree up to floating-point tolerance. Detectors with non-differentiable routing, top-k sparsification, or argmax pooling would require a separate reparameterization, whereas none of the evaluated baselines do.

Discrete consistency. The relaxation in Eq. (5) matches an edge flip at every binary vertex. For $q \in \{0,1\}^{M_n}$ and $(i,j) \in \mathcal{U}_n$,

$$\widetilde{A}_{ij}(q) = A_{ij} + (1 - 2A_{ij})q_{ij} = \begin{cases} A_{ij}, & q_{ij} = 0, \\ 1 - A_{ij}, & q_{ij} = 1, \end{cases}$$

Algorithm 1 SGC: Score-Directed Graph Cloaking

Input: trained score function s_θ, anomalous graph $G_a = (A, X)$, budget ratio ρ, step size η, iterations T, candidate factor c
Output: attacked binary graph $(A^\star, X)$

$k \leftarrow k_E(G_a; \rho)$
if $k = 0$ **then**
  return (A, X)
end if
$q \leftarrow \mathbf{0} \in [0,1]^{M_n}$
for $t = 0, \ldots, T-1$ **do**
  $\widetilde{A} \leftarrow \widetilde{A}(q)$ using Eq. (5)
  $g \leftarrow \nabla_q s_\theta(\widetilde{A}(q), X)$
  $\widehat{g} \leftarrow g \big/ \left(M_n^{-1} \sum_m |g_m| + \varepsilon_g\right)$
  $q \leftarrow \Pi_{\mathcal{C}^{\mathrm{atk}}_{G_a}(\rho)}(q - \eta \widehat{g})$
end for
$\mathcal{P} \leftarrow \mathrm{TopK}(q, \min\{M_n, \lceil ck \rceil\})$ ▷ candidate pool
$A_{\mathrm{cur}} \leftarrow A; \quad \mathcal{S} \leftarrow \emptyset$
for $\ell = 1, \ldots, k$ **do**
  $(i^\star, j^\star) \leftarrow \arg\min_{(i,j) \in \mathcal{P} \setminus \mathcal{S}} s_\theta(\mathrm{Flip}(A_{\mathrm{cur}}, i, j), X)$
  $A_{\mathrm{cur}} \leftarrow \mathrm{Flip}(A_{\mathrm{cur}}, i^\star, j^\star); \quad \mathcal{S} \leftarrow \mathcal{S} \cup \{(i^\star, j^\star)\}$
end for
if $s_\theta(A_{\mathrm{cur}}, X) < s_\theta(A, X)$ **then**
  $A^\star \leftarrow A_{\mathrm{cur}}$
else
  $A^\star \leftarrow A$ ▷ no-op fallback
end if
return $(A^\star, X)$

which is exactly the XOR operation in Eq. (3). Therefore the rounded output $A^\star$ is binary, symmetric, self-loop-free, and satisfies

$$\sum_{(i,j) \in \mathcal{U}_n} \mathbf{1}\{A^\star_{ij} \neq A_{ij}\} \leq k_E(G_a; \rho).$$

The final graph is always a valid discrete graph under the stated edit budget, regardless of the relaxed trajectory.

Gradient through the symmetric relaxation. Because a single upper-triangular variable controls both symmetric entries, the chain rule gives

$$\frac{\partial s_\theta}{\partial q_{ij}} = (1 - 2A_{ij}) \left(\frac{\partial s_\theta}{\partial \widetilde{A}_{ij}} + \frac{\partial s_\theta}{\partial \widetilde{A}_{ji}} \right), \qquad (i,j) \in \mathcal{U}_n. \tag{15}$$

Autograd yields this expression when $\widetilde{A}$ is constructed with explicit symmetric ties. We include Eq. (15) to make the insertion/deletion sign visible: the same negative score gradient can correspond to adding a missing edge or deleting an existing one, depending on A_{ij}.

Fixed protocol and restarts. The attack constants in Algorithm 1 are listed in Appendix C. We keep the main protocol zero-initialized and deterministic so that directedness gaps are comparable across detectors. Since the relaxed objective is non-convex, we separately audit under-optimization by increasing PGD steps and adding the restart configuration in Appendix C. These stronger attacks are diagnostics and do not change the main attack definition.

Relaxed-to-discrete diagnostic. SGC optimizes a relaxed graph but reports a binary graph. To check whether the relaxation is exploiting a non-discretizable weighted structure, we record

$$\Delta_{\mathrm{r}\to\mathrm{d}}(G_a) = s_\theta(A^\star, X) - s_\theta(\widetilde{A}(q^{(T)}), X). \tag{16}$$

Algorithm 2 AST: Adversarial Score Training

Input: training set $\mathcal{D}_{\mathrm{tr}}$, score function s_θ, base loss $\mathcal{L}_{\mathrm{base}}$, training budget ρ_{tr}, inner steps T_{in}, inner step size α, weight λ, floors $\varepsilon, \varepsilon_g$

for each minibatch $\mathcal{B} \subset \mathcal{D}_{\mathrm{tr}}$ **do**
  $\mathcal{J} \leftarrow 0$
  for all $G_b = (A_b, X_b) \in \mathcal{B}$ **do**
    $k_b \leftarrow k_E(G_b; \rho_{\mathrm{tr}})$; $M_b \leftarrow |\mathcal{U}_{n_b}|$
    $q \leftarrow \mathbf{0} \in [0,1]^{M_b}$; $q_{\mathrm{best}} \leftarrow q$; $s_{\mathrm{best}} \leftarrow s_\theta(A_b, X_b)$
    for $t = 1, \ldots, T_{\mathrm{in}}$ **do**
      $\widetilde{A} \leftarrow \widetilde{A}(q)$
      $g \leftarrow \nabla_q s_\theta(\widetilde{A}(q), X_b)$
      $\widehat{g} \leftarrow g / \left(M_b^{-1} \sum_m |g_m| + \varepsilon_g\right)$
      $q \leftarrow \Pi_{\mathcal{C}(k_b)}(q + \alpha \widehat{g})$
      if $s_\theta(\widetilde{A}(q), X_b) > s_{\mathrm{best}}$ **then**
        $q_{\mathrm{best}} \leftarrow q$; $s_{\mathrm{best}} \leftarrow s_\theta(\widetilde{A}(q), X_b)$
      end if
    end for
    $q_b^+ \leftarrow \mathrm{sg}(q_{\mathrm{best}})$ ▷ first-order outer update; $q = \mathbf{0}$ is feasible
    $s_0 \leftarrow s_\theta(A_b, X_b)$; $s_+ \leftarrow s_\theta(\widetilde{A}(q_b^+), X_b)$
    $\sigma_b \leftarrow \mathrm{sg}(\max\{|s_0|, \varepsilon\})$
    $\mathcal{R}_b \leftarrow ((s_+ - s_0)/\sigma_b)^2$
    $\mathcal{J} \leftarrow \mathcal{J} + \mathcal{L}_{\mathrm{base}}(\theta; G_b) + \lambda \mathcal{R}_b$
  end for
  update θ by descending $\nabla_\theta(\mathcal{J}/|\mathcal{B}|)$
end for

A large positive value would indicate that the relaxed optimizer found a score reduction that rounding could not realize. This diagnostic is not used to compute robustness metrics: all AUC, AUPRC, and ASR values are computed from the final binary $A^\star$.

B.2 Projected Inner Optimization for AST

Capped-simplex projection. Both SGC and AST repeatedly project onto a capped simplex. For $v \in \mathbb{R}^{M_n}$ and budget k, define

$$\Pi_{\mathcal{C}(k)}(v) = \arg \min_{q \in [0,1]^{M_n},\, \mathbf{1}^\top q \le k} \|q - v\|_2^2.$$

If $\mathbf{1}^\top \mathrm{clip}(v, 0, 1) \le k$, the projection is simply $\mathrm{clip}(v, 0, 1)$. Otherwise, the KKT conditions give

$$\left[\Pi_{\mathcal{C}(k)}(v)\right]_m = \min\{1, \max\{0, v_m - \nu\}\}, \tag{17}$$

where ν is chosen so that $\sum_m [\Pi_{\mathcal{C}(k)}(v)]_m = k$. We compute ν with a sorted-pivot routine in $O(M_n \log M_n)$ time [13]. We use the Euclidean projection rather than multiplicative rescaling: rescaling is feasible in some cases, but it is not the projection, biases all coordinates uniformly, and does not correctly handle box violations.

Gradient normalization. The inner ascent in AST uses the same mean-absolute normalization as SGC:

$$\widehat{g} = \frac{\nabla_q s_\theta(\widetilde{A}(q), X)}{M_n^{-1} \sum_{(i,j) \in \mathcal{U}_n} \left| \partial s_\theta(\widetilde{A}(q), X)/\partial q_{ij} \right| + \varepsilon_g}. \tag{18}$$

This preserves the relative ranking of candidate edits while removing global score-scale effects. The value of ε_g and all other attack/defense constants are given in Appendix C.

Normal-only objective. If the inner maximization in Eq. (12) were exact, AST would penalize the empirical normal-side variation

$$\widehat{L}^2_{\mathrm{AST}}(\theta) = \frac{1}{|\mathcal{D}_{\mathrm{tr}}|} \sum_{G=(A,X)\in\mathcal{D}_{\mathrm{tr}}} \left[\sup_{q\in\mathcal{C}_G(\rho_{\mathrm{tr}})} \frac{s_\theta(\widetilde{A}(q),X) - s_\theta(A,X)}{\max\{|s_\theta(A,X)|,\varepsilon\}} \right]^2. \tag{19}$$

With finite projected ascent, the training loop optimizes a tractable lower-bound estimate of this supremum, because the generated q_b^+ is feasible but need not be globally optimal. The loss never uses anomalous examples, so it cannot be interpreted as a certificate on anomaly-side cloaking trajectories. This is why robustness must be tested with an external attack stronger than the training inner loop.

Inner-loop tightness diagnostic. To quantify how much variation the short training inner loop captures, we compare it with a stronger held-out solver. Let q^{train} be the best iterate returned by the T_{in}-step training inner solver, and let q^{strong} be the best perturbation found by the stronger diagnostic solver together with q^{train}. For a held-out normal graph $G = (A, X)$, define

$$r(G) = \begin{cases} \dfrac{s_\theta(\widetilde{A}(q^{\mathrm{train}}),X) - s_\theta(A,X)}{s_\theta(\widetilde{A}(q^{\mathrm{strong}}),X) - s_\theta(A,X)}, & s_\theta(\widetilde{A}(q^{\mathrm{strong}}),X) > s_\theta(A,X), \\ 1, & s_\theta(\widetilde{A}(q^{\mathrm{strong}}),X) = s_\theta(A,X), \end{cases} \in [0,1]. \tag{20}$$

A high ratio means that the training-time solver already captures most of the score-increasing variation found by the stronger solver. The value $r(G) = 1$ in the zero-denominator case means that neither solver found a feasible score-increasing perturbation.

B.3 Robustness Diagnostics

Gradient-masking check. Adversarial defenses can appear robust by obscuring gradients rather than by improving the score landscape [2]. We therefore evaluate AST-trained detectors with a stronger SGC configuration, including the attack budget, step, and restart settings listed in Appendix C. Along each SGC trajectory, we record $\|\nabla_q s_\theta(\widetilde{A}(q^{(t)}),X)\|_2$. A masking-like defense would typically show vanishing gradients at initialization, unstable gradient trajectories, or a sharp failure under stronger attacks. A genuinely smoother score landscape should instead show usable gradients whose magnitudes decay smoothly as the attack descends.

Encoder Lipschitz probe. AST's mechanism is indirect: smoothing is imposed on normal graphs, while cloaking is evaluated on anomalies. To probe whether the encoder propagates this smoothness, let ϕ_θ be the graph encoder and estimate, for graphs with $k_E(G;\rho) > 0$,

$$\widehat{L}_{\mathrm{enc}}(G;\rho) = \mathbb{E}_{q\sim p_\rho} \left[\frac{\|\phi_\theta(\widetilde{A}(q),X) - \phi_\theta(A,X)\|_2}{\|q\|_1} \right], \tag{21}$$

where p_ρ samples K budget-respecting upper-triangular masks with $k_E(G;\rho)$ active entries, and we use $K = 20$ Monte Carlo samples. We average this statistic on normal training graphs, held-out normal graphs, and anomalous test graphs. A reduction in anomalies supports the transfer hypothesis, whereas a reduction only in normal graphs would suggest that the defense improvement arises from another mechanism, such as generic weight regularization, rather than propagated structural smoothness.

Table 4: Dataset statistics for the five graph-level anomaly-detection benchmarks. "Node feat." is the input node-feature dimensionality after preprocessing, and "Avg. nodes" and "Avg. edges" are per-graph averages, which determine the scale of the edge-flip budget.

Dataset	#Graphs	#Nodes	#Node feat.	#Avg. nodes	#Avg. edges
AIDS	2,000	31,385	38	15.7	16.2
PROTEINS	1,113	43,471	3	39.1	72.8
COX2	467	19,252	35	41.2	43.4
ENZYMES	600	19,580	3	32.6	62.1
BZR	405	14,479	53	35.8	38.4

Table 5: Experimental protocol and hyperparameters for split construction, threshold calibration, test-time perturbation, SGC, AST, and robustness diagnostics. Unless stated otherwise, values are shared across all datasets and detectors.

Component	Setting	Value / choice
Data split	Stratified folds	5
	Validation-normal split	10% of the normal training graphs per fold
	Threshold	Validation-normal 95th percentile
Evaluation	Main edit budget	$\rho = 0.10$
	Budget-sensitivity sweep	$\rho \in \{0, 0.05, 0.10, 0.20\}$ on AIDS
	Random control repeats	$R = 5$ random repeats per fold
	Perturbed graphs	Anomalous test graphs only with normal test graphs unchanged
SGC attack	PGD steps	$T = 100$ for main evaluation
	Step size / gradient floor	$\eta = 0.1$, $\varepsilon_g = 10^{-12}$
	Initialization / restarts	$q^{(0)} = \mathbf{0}$. No random restarts in the main protocol
	Candidate pool	Top-min$\{M_n, \lceil c\, k_E(G;\rho)\rceil\}$ with $c = 5$
AST training	Inner ascent	$T_{\text{in}} = 5$, $\rho_{\text{tr}} = 0.10$, normalized-gradient ascent
	Defense parameter	$\alpha = 0.1$, $\lambda = 1000$ as their default values
	Normalization floors	$\varepsilon = 1$, $\varepsilon_g = 10^{-12}$
Diagnostics	Strong AST inner solver	$T_{\text{in}}^{\text{strong}} = 50$ with 3 restarts initialized as $q^{(0)} \sim \text{Uniform}([0, 0.1]^{M_n})$
	Strong SGC audit	$T = 200$ with 3 random restarts when used; diagnostic budgets specified in each experiment
	Encoder Lipschitz probe	$K = 20$ Monte Carlo masks

C More Experimental Details

Dataset description. Here, we provide a brief overview of the dataset statistics used in our experiment, which are shown in Table 4.

Split construction and threshold calibration. We follow the UB-GOLD [37] graph-level anomaly detection protocol, which treats the first class of each dataset as anomalous, and the other classes as normal. We use five stratified folds, and within each fold, we reserve 10% of the normal training graphs for threshold calibration and use the remaining normal graphs for training. For each trained detector, τ is set to the 95th percentile of its oriented anomaly scores on this held-out subset and remains fixed across clean and attacked evaluations. Ranking metrics use oriented scores on the full test fold, while threshold-based metrics use τ. Table 5 summarizes the split, calibration, perturbation budget, attack, defense, and diagnostic settings.

Test-time evaluation protocols. We compare three conditions. CLEAN: the original test graphs without perturbation. RS (random-structure perturbation): samples $k_E(G_a;\rho)$ unordered node pairs uniformly without replacement and flips them under the same budget as the attack. Results are averaged over $R = 5$ random repeats. SGC: a white-box score-directed attack in Algorithm 1 optimizes a relaxed dense adjacency and greedily rounds the result to a discrete graph. In both RS and SGC, only anomalous test graphs are perturbed, normal test graphs remain unchanged, and node features, labels, training data, and trained model parameters are fixed. Unless otherwise stated, the evaluation budget is $\rho = 0.10$.

Computing Resources All experiments were conducted on the Linux server equipped with NVIDIA Tesla H100 GPUs with 80GB memory and Intel Xeon Platinum 8480CL CPUs. Unless otherwise stated, each training, attack, or diagnostic run used a single GPU. Runtime varied across datasets and detectors due to graph size, number of anomalous test graphs, and attack cost. A five-fold vanilla dataset–model run, including training, clean evaluation, Random-Structure evaluation, and SGC evaluation at $\rho = 0.10$, typically required several GPU hours. Budget sweeps scaled roughly linearly with the number of tested budgets, while score-distribution, margin-decrease, and gradient diagnostics

reused trained detectors when possible and were mainly attack/evaluation costs. Overall, the reported experiments required on the order of hundreds of single-GPU hours.

Realized structural edits. We count the final discrete edits over all anomalous test graphs submitted to SGC, including successful and unsuccessful attacks. One edit inserts or deletes one undirected edge. At $\rho = 0.10$, the mean $\pm$ standard deviation is 3.61 ± 0.33 on AIDS, 8.32 ± 1.45 on PROTEINS, 3.00 ± 1.40 on COX2, 5.86 ± 0.71 on ENZYMES, and 3.19 ± 0.48 on BZR. These counts describe the attacked anomalous graphs, whose size distribution can differ from the full-dataset averages in Table 4. The minimum budget of one edit in Eq. (2) also makes the realized fraction graph-dependent at small graph sizes.

Attack complexity and runtime. The dense implementation stores $M = n(n-1)/2$ candidate edit variables, giving $O(n^2)$ attack-specific memory in addition to the detector's memory. Let k be the edit budget and $P = \min\{M, \lceil ck \rceil\}$ the rounding pool size. If $C_{\mathrm{fb}}(n)$ and $C_{\mathrm{fwd}}(n)$ denote the costs of one detector forward-backward pass and one forward pass, respectively, the per-graph attack cost is

$$O\big(T\big(C_{\mathrm{fb}}(n) + M \log M\big) + kP\, C_{\mathrm{fwd}}(n)\big)\,.$$

The first term covers projected descent with sorted-pivot projection, and the second covers greedy evaluation of candidate flips. We measure OCGIN attack time on 20 randomly sampled anomalous graphs per dataset from AIDS, COX2, ENZYMES, and PROTEINS, using $T = 100$, $\rho = 0.10$, and $c = 5$. Dataset-level mean times, including descent and rounding, range from 0.523 to 0.702 seconds per graph. The largest sampled graph has 149 nodes and takes 2.64 seconds. These measurements characterize the small and medium graphs in the timing sample. The quadratic candidate space remains the main memory constraint as graph size increases, while attacking additional graphs incurs an additive cost.

D Additional Experimental Results

D.1 Evaluations on More Datasets

In this subsection, we further conduct the attack experiment on more datasets, including COX2 and BZR. Tables 6 and 7 summarize the experimental results, where we can observe the consistent trend as in Table 1.

Table 6: Additional AUC results for undefended GLAD models under random-structure (RS) and SGC perturbations. The protocol is identical to Table 1.

Dataset	Model	AUC↓			ASR ↑		DG ↑
		Clean	RS	SGC	RS	SGC	
COX2	OCGIN	0.526±0.099	0.813±0.111	**0.446±0.118**	0.000±0.000	**0.286±0.203**	0.367
	GLADC	0.612±0.097	0.899±0.042	**0.053±0.037**	0.027±0.060	**1.000±0.000**	0.846
	GLocalKD	0.549±0.061	0.550±0.062	**0.457±0.052**	0.102±0.087	**0.367±0.164**	0.093
	MUSE	0.713±0.065	0.949±0.034	**0.385±0.354**	0.000±0.000	**0.600±0.548**	0.564
	GLADPro	0.727±0.056	0.843±0.056	**0.023±0.015**	0.115±0.118	**1.000±0.000**	0.820
BZR	OCGIN	0.646±0.048	0.803±0.053	**0.545±0.052**	0.078±0.076	**0.659±0.202**	0.258
	GLADC	0.579±0.038	0.872±0.026	**0.136±0.071**	0.004±0.009	**0.979±0.047**	0.736
	GLocalKD	0.659±0.081	0.645±0.082	**0.526±0.098**	0.078±0.021	**0.291±0.110**	0.119
	MUSE	0.778±0.068	0.913±0.049	**0.326±0.279**	0.060±0.089	**0.876±0.172**	0.587
	GLADPro	0.764±0.052	0.609±0.096	**0.090±0.053**	0.400±0.177	**0.982±0.017**	0.519

D.2 Clean–Robust Alignment under AST

A natural concern with a robustness regularizer is that improved attacked performance may come from sacrificing clean detection. To answer this question, we examine this possibility by measuring the joint change in clean AUC and SGC AUC induced by AST. Figure 5 plots ΔSGC AUC against Δ Clean AUC for all 25 dataset–model pairs. We can notice that most pairs move upward in robust AUC while maintaining comparable clean AUC, although some of the largest robustness improvements accompany small clean-AUC drops. These experimental results suggest that AST

Table 7: Additional results for adversarial score training (AST) under SGC. The protocol is identical to Table 3.

Dataset	Model	Training	Clean AUC ↑	SGC AUC ↑	SGC ASR ↓	Δ SGC AUC ↑	Δ SGC ASR ↓
COX2	OCGIN	Vanilla	0.526±0.099	0.446±0.118	0.286±0.203	+0.058	-0.108
		AST	**0.587±0.083**	**0.504±0.139**	**0.178±0.143**		
	GLADC	Vanilla	0.612±0.097	0.053±0.037	1.000±0.000	+0.194	-0.288
		AST	**0.649±0.123**	**0.247±0.039**	**0.712±0.085**		
	GLocalKD	Vanilla	0.549±0.061	0.457±0.052	0.367±0.164	+0.021	-0.145
		AST	**0.553±0.060**	**0.478±0.056**	**0.222±0.118**		
	MUSE	Vanilla	**0.713±0.065**	0.385±0.354	0.600±0.548	+0.174	-0.400
		AST	0.705±0.073	**0.559±0.319**	**0.200±0.347**		
	GLADPro	Vanilla	**0.727±0.056**	0.023±0.015	1.000±0.000	+0.323	-0.372
		AST	0.713±0.051	**0.346±0.039**	**0.628±0.142**		
BZR	OCGIN	Vanilla	0.646±0.048	0.545±0.052	0.659±0.202	+0.058	-0.221
		AST	**0.655±0.063**	**0.603±0.075**	**0.438±0.205**		
	GLADC	Vanilla	0.579±0.038	0.136±0.071	0.979±0.047	+0.281	-0.337
		AST	**0.650±0.053**	**0.417±0.130**	**0.642±0.136**		
	GLocalKD	Vanilla	**0.659±0.081**	0.526±0.098	0.291±0.110	+0.026	-0.075
		AST	0.653±0.082	**0.552±0.095**	**0.216±0.091**		
	MUSE	Vanilla	**0.778±0.068**	0.326±0.279	0.876±0.172	+0.226	-0.314
		AST	0.758±0.079	**0.552±0.118**	**0.562±0.137**		
	GLADPro	Vanilla	**0.764±0.052**	0.090±0.053	0.982±0.017	+0.262	-0.364
		AST	0.743±0.081	**0.352±0.088**	**0.618±0.099**		

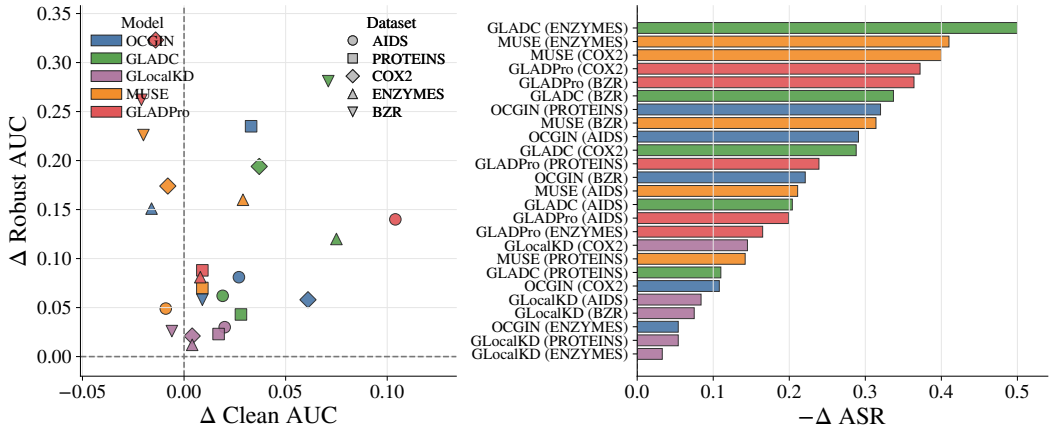


Figure 5: Clean–robust AUC alignment under AST across all 25 dataset–model pairs from Tables 3 and 7. The x-axis reports Δ Clean AUC, and the y-axis reports Δ SGC AUC, both computed as AST minus Vanilla. In the left panel, marker color denotes the model and marker shape denotes the dataset. The right panel ranks dataset–model pairs by ASR reduction, computed as Vanilla ASR minus AST ASR under SGC.

improves cloaking robustness while largely preserving clean ranking performance, which fully demonstrates its effectiveness.

D.3 Evaluation with AUPRC under SGC and AST

Since graph anomaly splits can be imbalanced, we also report AUPRC as a prevalence-sensitive ranking metric. Table 8 shows the AUPRC performance of all baselines across five datasets, where we also report the directedness gap (DG) for AUPRC. We can observe that DG is positive across all cases. Thus, the directedness gap is not an artifact of AUROC, and score-directed edits also degrade the precision-recall ranking of anomalies relative to budget-matched random edits.

In addition, we report the AUPRC results under our defense experiment with AST, where the results are summarized in Table 9. We observe that AST also improves robustness, as measured by AUPRC, across all cases. This further qualifies the defense claim in the right direction: AST is an effective empirical robustness baseline.

Table 8: AUPRCs of undefended GLAD models under random-structure (RS) and SGC perturbations. Directedness gap (DG) is the AUPRC gap between RS and SGC. Note that lower AUPRC here denotes the stronger attack effectiveness. A larger value for DG indicates stronger score-directed vulnerability beyond random graph noise.

Dataset	Model	AUPRC↓			DG ↑
		Clean	RS	SGC	
AIDS	OCGIN	0.913±0.035	0.902±0.036	**0.787±0.060**	0.115
	GLADC	0.857±0.051	0.921±0.038	**0.641±0.067**	0.280
	GLocalKD	0.924±0.034	0.918±0.034	**0.889±0.037**	0.029
	MUSE	0.787±0.042	0.830±0.049	**0.396±0.029**	0.434
	GLADPro	0.331±0.031	0.511±0.139	**0.147±0.025**	0.364
PROTEINS	OCGIN	0.793±0.021	0.759±0.024	**0.574±0.011**	0.185
	GLADC	0.814±0.039	0.866±0.030	**0.651±0.041**	0.215
	GLocalKD	0.819±0.020	0.810±0.019	**0.779±0.022**	0.031
	MUSE	0.741±0.047	0.842±0.043	**0.532±0.035**	0.310
	GLADPro	0.769±0.020	0.838±0.017	**0.617±0.058**	0.221
ENZYMES	OCGIN	0.296±0.080	0.261±0.088	**0.136±0.024**	0.125
	GLADC	0.203±0.047	0.310±0.073	**0.114±0.014**	0.196
	GLocalKD	0.271±0.075	0.257±0.073	**0.223±0.059**	0.034
	MUSE	0.282±0.030	0.382±0.086	**0.128±0.024**	0.254
	GLADPro	0.250±0.042	0.313±0.079	**0.123±0.018**	0.190
COX2	OCGIN	0.795±0.063	0.907±0.057	**0.706±0.072**	0.201
	GLADC	0.827±0.052	0.963±0.017	**0.591±0.021**	0.372
	GLocalKD	0.801±0.046	0.802±0.046	**0.708±0.043**	0.094
	MUSE	0.889±0.028	0.982±0.017	**0.741±0.160**	0.241
	GLADPro	0.890±0.019	0.947±0.023	**0.609±0.007**	0.338
BZR	OCGIN	0.872±0.028	0.928±0.021	**0.702±0.034**	0.226
	GLADC	0.825±0.019	0.962±0.010	**0.619±0.023**	0.343
	GLocalKD	0.876±0.038	0.868±0.040	**0.734±0.051**	0.134
	MUSE	0.905±0.047	0.968±0.022	**0.708±0.122**	0.260
	GLADPro	0.931±0.018	0.877±0.030	**0.620±0.019**	0.257

D.4 Sensitivity on the Step of SGC and Restart Strategy

In this section, we evaluate whether the main SGC configuration is under-optimized by increasing the number of PGD steps or adding random restarts. Specifically, we compare the main setting ($T = 100$, zero initialization), a longer setting ($T = 200$), and a stronger restart setting ($T = 200$ with three random initializations) on the AIDS dataset. All settings follow the same training protocol and set $\rho = 0.10$. Table 10 shows the experimental results. We can observe that the stronger configurations still preserve the main conclusions. Increasing T from 100 to 200 has negligible effects on GLocalKD and MUSE, and only small changes on OCGIN and GLADPro. In particular, GLADC performs the same in all three cases. Adding restarts further strengthens the attack in several cases, most visibly for GLADPro, but the changes are incremental relative to the main-table directedness gaps. The main $T = 100$ setting is therefore not materially under-optimized: stronger optimization leaves the detector ordering and the substantive cloaking conclusions unchanged.

D.5 Gradient-Masking Diagnostic for AST

In this section, we check whether the apparent robustness gains of AST could be explained by gradient masking. A masked defense can make a white-box attack appear weak by producing vanishing or unusable gradients, rather than by making the score surface genuinely harder to exploit. We therefore run a stronger SGC attack on AIDS with $\rho = 0.10$ and $T = 200$, and record the gradient of the relaxed anomaly score with respect to the continuous edge-flip variables.

Table 9: Effect of adversarial score training (AST) under SGC in terms of AUPRC. All results are reported as five-fold mean ± standard deviation. Δ SGC AUPRC denotes AST minus Vanilla for the same dataset–model pair. Positive Δ SGC AUPRC indicates improved robustness.

Dataset	Model	Training	Clean AUPRC ↑	SGC AUPRC ↑	Δ SGC AUPRC ↑
AIDS	OCGIN	Vanilla	0.913±0.035	0.787±0.060	+0.209
		AST	**0.997±0.004**	**0.996±0.004**	
	GLADC	Vanilla	0.857±0.051	0.641±0.067	+0.204
		AST	**0.956±0.036**	**0.845±0.067**	
	GLocalKD	Vanilla	0.924±0.034	0.889±0.037	+0.088
		AST	**0.987±0.008**	**0.977±0.010**	
	MUSE	Vanilla	**0.787±0.042**	0.396±0.029	+0.107
		AST	0.760±0.014	**0.503±0.030**	
	GLADPro	Vanilla	0.331±0.031	0.147±0.025	+0.082
		AST	**0.479±0.094**	**0.229±0.015**	
PROTEINS	OCGIN	Vanilla	0.793±0.021	0.574±0.011	+0.161
		AST	**0.816±0.016**	**0.735±0.041**	
	GLADC	Vanilla	0.814±0.039	0.651±0.041	+0.064
		AST	**0.840±0.018**	**0.715±0.019**	
	GLocalKD	Vanilla	0.819±0.020	0.779±0.022	+0.026
		AST	**0.835±0.018**	**0.805±0.017**	
	MUSE	Vanilla	0.741±0.047	0.532±0.035	+0.059
		AST	**0.763±0.034**	**0.591±0.034**	
	GLADPro	Vanilla	**0.769±0.020**	0.617±0.058	+0.044
		AST	0.732±0.079	**0.661±0.023**	
ENZYMES	OCGIN	Vanilla	**0.296±0.080**	0.136±0.024	+0.038
		AST	0.283±0.076	**0.174±0.033**	
	GLADC	Vanilla	0.203±0.047	0.114±0.014	+0.039
		AST	**0.238±0.053**	**0.153±0.016**	
	GLocalKD	Vanilla	0.271±0.075	0.223±0.059	+0.036
		AST	**0.281±0.063**	**0.259±0.055**	
	MUSE	Vanilla	0.282±0.030	0.128±0.024	+0.135
		AST	**0.331±0.105**	**0.263±0.048**	
	GLADPro	Vanilla	0.250±0.042	0.123±0.018	+0.061
		AST	**0.287±0.036**	**0.184±0.007**	
COX2	OCGIN	Vanilla	0.795±0.063	0.706±0.072	+0.070
		AST	**0.819±0.057**	**0.776±0.076**	
	GLADC	Vanilla	0.827±0.052	0.591±0.021	+0.038
		AST	**0.847±0.056**	**0.629±0.011**	
	GLocalKD	Vanilla	**0.801±0.046**	0.708±0.043	+0.064
		AST	0.799±0.047	**0.772±0.045**	
	MUSE	Vanilla	**0.889±0.028**	0.741±0.160	+0.071
		AST	0.875±0.049	**0.812±0.136**	
	GLADPro	Vanilla	**0.890±0.019**	0.609±0.007	+0.014
		AST	0.883±0.028	**0.623±0.005**	
BZR	OCGIN	Vanilla	0.872±0.028	0.702±0.034	+0.100
		AST	**0.878±0.039**	**0.802±0.063**	
	GLADC	Vanilla	0.825±0.019	0.619±0.023	+0.068
		AST	**0.852±0.020**	**0.687±0.044**	
	GLocalKD	Vanilla	**0.876±0.038**	0.734±0.051	+0.073
		AST	0.872±0.039	**0.807±0.054**	
	MUSE	Vanilla	**0.905±0.047**	0.708±0.122	+0.098
		AST	0.894±0.054	**0.806±0.071**	
	GLADPro	Vanilla	**0.931±0.018**	0.620±0.019	+0.099
		AST	0.926±0.036	**0.719±0.041**	

Table 10: Sensitivity analysis of step and restart numbers of SGC with $\rho = 0.10$. We compare the main attack setting ($T = 100$), a longer attack ($T = 200$), and $T = 200$ with three random restarts. Lower SGC AUC, higher SGC ASR indicate a stronger attack.

Model	Attack setting	SGC AUC ↓	SGC ASR ↑
OCGIN	$T = 100$	0.916 ± 0.032	0.294 ± 0.050
	$T = 200$	0.914 ± 0.030	0.276 ± 0.053
	$T = 200$ + 3 restarts	0.897 ± 0.039	0.317 ± 0.059
GLADC	$T = 100$	0.886 ± 0.024	0.448 ± 0.084
	$T = 200$	0.886 ± 0.024	0.448 ± 0.084
	$T = 200$ + 3 restarts	0.886 ± 0.024	0.448 ± 0.084
GLocalKD	$T = 100$	0.957 ± 0.019	0.104 ± 0.044
	$T = 200$	0.957 ± 0.018	0.111 ± 0.030
	$T = 200$ + 3 restarts	0.954 ± 0.019	0.130 ± 0.039
MUSE	$T = 100$	0.793 ± 0.021	0.835 ± 0.079
	$T = 200$	0.789 ± 0.023	0.842 ± 0.075
	$T = 200$ + 3 restarts	0.783 ± 0.024	0.855 ± 0.069
GLADPro	$T = 100$	0.322 ± 0.118	0.923 ± 0.084
	$T = 200$	0.305 ± 0.110	0.903 ± 0.066
	$T = 200$ + 3 restarts	0.261 ± 0.108	0.980 ± 0.045

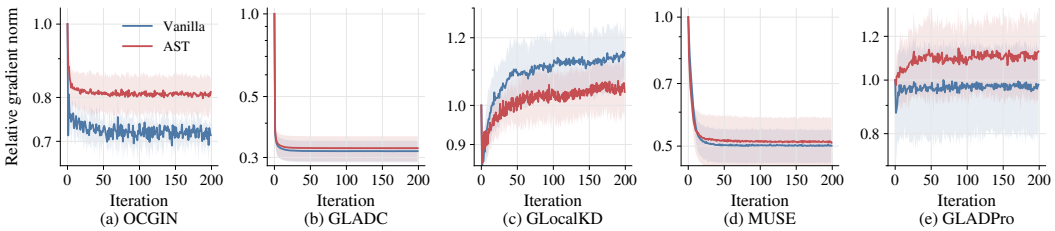


Figure 6: **Gradient-masking diagnostic for AST.** We implemented a stronger SGC with $\rho = 0.10$ and $T = 200$ and show the relative gradient norm along the relaxed attack trajectory. Each curve is normalized by its initial iteration value within the same run. Note that the lines denote five-fold means and shaded regions denote standard deviation.

Figure 6 shows the normalized gradient trajectories of all five GLAD detectors on AIDS. For each run, the gradient norm at iteration t is divided by its value at the initial iteration, so the experiment tests whether gradients collapse during optimization while avoiding raw score-scale differences across detectors. We observe that the AST curves remain bounded away from zero across all five detectors. In some cases, such as GLADPro, they are comparable to or larger than the Vanilla curves. Thus, the observed improvements are not explained by the most direct vanishing-gradient form of gradient masking. This diagnostic is not a certificate against all forms of gradient obfuscation. Instead, it shows that the external AST optimizer continues to receive usable gradients under SGC.

D.6 Independent Audits and a Cloaking Case

The audit evaluates whether graphs that evade OCGIN retain their anomalous designation under models trained independently of the attack target. The supervised auditor f_{ext} is a GIN classifier trained on the original AIDS class labels and achieves clean-validation AUC 0.972 ± 0.016 across five folds. Its output is mapped to the benchmark's normal and anomalous classes. GLADC and GLocalKD provide two additional checks using their own anomaly scores and validation-normal thresholds. All three models are queried only after the edited graphs have been generated.

For unconstrained SGC, f_{ext} retains the anomalous class for 86 of the 104 OCGIN evasions, corresponding to 82.7% retention. Counting these audited evasions against the original clean-detected subset gives audited ASR 0.243 ± 0.052, compared with raw ASR 0.294 ± 0.050. Under the structural constraints in Section 3.3, f_{ext} retains 44 of the 48 SGC evasions, GLADC flags 45, and GLocalKD flags 44. Their intersection contains 40 evasions. The joint criterion therefore combines

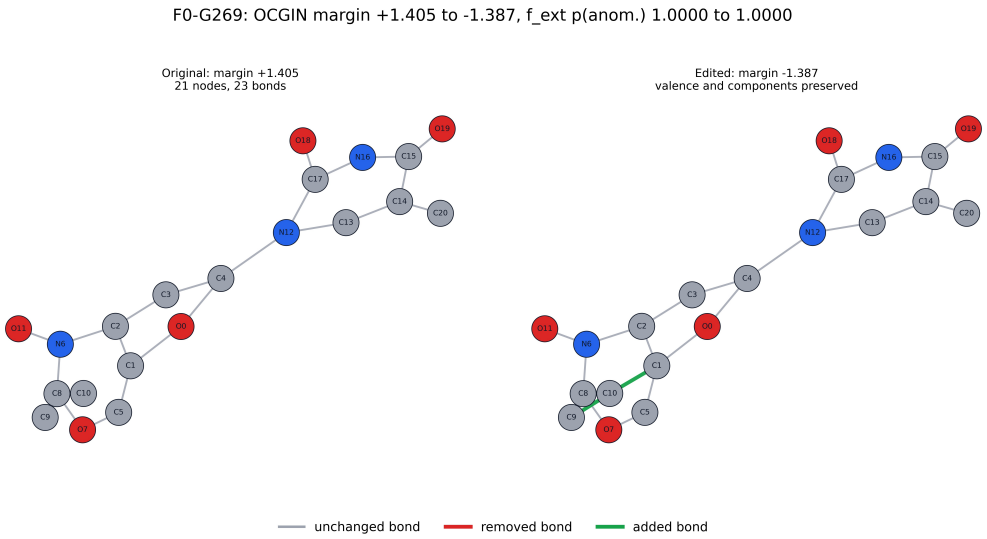


Figure 7: An independently audited cloaking case on AIDS. Atom labels combine the element and node index. Green edges mark the two insertions, and gray edges are unchanged. The OCGIN margin is the anomaly score minus its validation-normal threshold. Both graphs retain the anomalous designation under f_{ext}, GLADC, and GLocalKD.

Table 11: Attack comparison on AIDS with OCGIN at $\rho = 0.10$. Results are five-fold means $\pm$ standard deviations. All attacks share the same structural edit space and budget.

Attack	Guidance	ASR ↑	AUC ↓
RS	Random selection	0.067 ± 0.026	0.968 ± 0.015
FD-50	Up to 50 score probes	0.236 ± 0.056	0.926 ± 0.029
One-step saliency	One clean-graph gradient	0.249 ± 0.079	0.925 ± 0.024
SGC	100 projected-gradient steps	0.294 ± 0.050	0.916 ± 0.032

agreement with an external label predictor and agreement across two separately learned notions of abnormality. Together with the matched constrained RS control, these checks support the persistence of detector-specific evasion under the stated edit rules.

Representative case. Figure 7 shows graph F0-G269, where the identifier records fold 0 and graph 269. The original graph has 21 nodes and 23 edges. SGC adds the edges C1-C10 and C9-C10, preserving the maximum-valence constraints and the number of connected components. These two edits move the OCGIN margin from $+1.405$ to -1.387, crossing the detection threshold. The external classifier's anomaly probability remains 1.000 to the reported precision, and both GLADC and GLocalKD continue to flag the graph. This case illustrates how a small localized connectivity change can alter the target detector's decision while the independently audited anomaly indicators remain stable.

D.7 Additional Score-Guided Attack Baselines

Here, we compare different forms of score guidance approaches. Specifically, we evaluate OCGIN on AIDS by holding the edit space and budget $\rho = 0.10$ fixed. The one-step score-saliency attack uses the gradient at the clean graph to rank edge flips by their predicted decrease in anomaly score. The query-limited attack FD-50 evaluates up to 50 one-edge perturbations, estimates their score changes by finite differences, and selects a budget-feasible set of score-decreasing candidates. It uses approximately 52 scalar-score queries per graph, including the clean and final evaluations. SGC uses the main configuration of 100 projected-gradient steps followed by greedy rounding.

Table 11 shows that both FD-50 and one-step saliency substantially increase mean ASR over RS, from 0.067 to 0.236 and 0.249, respectively. Their similar post-attack AUCs indicate that useful

Table 12: Scale-normalized encoder sensitivity for OCGIN. Values are five-fold means $\pm$ standard deviations on clean normal and anomalous test graphs, with perturbations sampled at $\rho = 0.10$.

Dataset	Test subset	Vanilla	AST
AIDS	Normal	0.788 ± 0.011	0.244 ± 0.021
AIDS	Anomalous	0.224 ± 0.022	0.055 ± 0.015
ENZYMES	Normal	0.168 ± 0.045	0.105 ± 0.027
ENZYMES	Anomalous	0.193 ± 0.008	0.128 ± 0.006

Table 13: Threshold calibration on clean AIDS test graphs. Each model uses its own validation-normal 95th-percentile threshold. Values are five-fold means $\pm$ standard deviations.

Detector	Training	Normal-test FPR	Clean-anomaly TPR
OCGIN	Vanilla	0.061 ± 0.026	0.885 ± 0.085
OCGIN	AST	0.051 ± 0.023	0.993 ± 0.011
GLADC	Vanilla	0.059 ± 0.019	0.865 ± 0.047
GLADC	AST	0.046 ± 0.018	0.960 ± 0.049

evasion directions can be identified through either a limited set of discrete score queries or a single local gradient. Iterative SGC further raises mean ASR to 0.294 and lowers mean AUC to 0.916. The comparison connects attack strength to the available score guidance: limited feedback already exposes substantial vulnerability, while iterative optimization yields additional gains in this setting.

D.8 Additional Validation of AST

We examine AST from four complementary perspectives: sensitivity of the learned representation, training inner-loop tightness, calibration on clean normal graphs, and robustness to an additional attack objective. The encoder diagnostics use OCGIN on AIDS and ENZYMES. Calibration is evaluated for OCGIN and GLADC on AIDS, and the adaptive attack targets OCGIN on AIDS. All structural probes and attacks in this subsection use $\rho = 0.10$.

Encoder sensitivity on normal and anomalous graphs. We evaluate the probe in Eq. (21) with $K = 20$ sampled masks and normalize its value by $\|\phi_\theta(A, X)\|_2$, the clean embedding norm from the same model. This normalization accounts for the different embedding scales of separately trained Vanilla and AST encoders. The resulting statistic measures sampled relative embedding displacement per edge edit, with smaller values indicating lower local sensitivity.

Table 12 shows lower normalized sensitivity under AST on both test subsets. On AIDS, the reductions are approximately 69% for normal graphs and 75% for anomalies. On ENZYMES, they are approximately 38% and 34%, respectively. The comparable reductions within each dataset support the hypothesis that the representation changes associated with normal-side smoothing extend to anomalous inputs. The smaller reductions on ENZYMES also indicate that the extent of this transfer depends on the dataset.

Training inner-loop tightness. On normal test graphs, we compare the five-step training solver with a diagnostic solver using 50 steps and three restarts, following Eq. (20). The mean ratio increases from 0.650 under Vanilla to 0.748 under AST on AIDS, and from 0.501 to 0.634 on ENZYMES. Thus, the short solver recovers a larger fraction of the score increase found by the stronger search after AST training. This result concerns agreement between the two finite solvers and complements the encoder sensitivity measurements.

Normal-test calibration. For each trained model, we calibrate the threshold using the 95th percentile of its own validation-normal scores. We then measure FPR on clean normal test graphs and TPR on clean anomalous test graphs. This evaluates how AST changes detection behavior at the same nominal validation-normal operating point.

Table 13 shows that AST maintains mean normal-test FPR close to the nominal 5% level while improving clean-anomaly TPR for both detectors. OCGIN's TPR increases from 0.885 to 0.993, and

Table 14: Adaptive attacks on AIDS with OCGIN at $\rho = 0.10$. Pooled ASR reports total evasions divided by the clean-detected anomaly count for each trained model. AUC values are five-fold means $\pm$ standard deviations. The adaptive attack selects the lower-score candidate from the score and representation objectives.

Training	Attack	Pooled ASR	AUC ↑
Vanilla	SGC	$0.294\ (104/354)$	0.916 ± 0.032
Vanilla	Adaptive	$0.331\ (117/354)$	0.908 ± 0.033
AST	SGC	$0.003\ (1/397)$	0.997 ± 0.004
AST	Adaptive	$0.005\ (2/397)$	0.996 ± 0.006

GLADC's from 0.865 to 0.960. These results place the defense gains at a comparable normal-test false-positive level and show improved detection of clean anomalies at that operating point.

Adaptive attack with a representation objective. We supplement scalar-score minimization with an objective that minimizes the squared distance between the edited graph embedding and the centroid of normal embeddings. Each objective produces a candidate graph under the same edge-edit budget. For each input, the attacker retains the candidate with the lower final anomaly score. This selection rule tests whether movement toward the normal representation centroid exposes additional score-decreasing directions in the deployed model.

The representation objective increases Vanilla evasions from 104 to 117 and lowers AUC from 0.916 to 0.908, demonstrating that it contributes additional attack candidates beyond the original SGC objective. Under AST, the same selection rule increases evasions from one to two, with AUC remaining at 0.996. Thus, the defense improvement persists under this additional search objective on AIDS with OCGIN. ASR uses each model's own clean-detected subset, containing 354 anomalies for Vanilla and 397 for AST, while AUC measures ranking performance on the full test sets.

E Broader Impact

This work aims to improve the reliability of unsupervised graph-level anomaly detection by revealing a gap between clean-test performance and robustness under test-time graph cloaking. Its positive impact lies in promoting more realistic robustness evaluation for graph anomaly detectors used in security-sensitive domains such as fraud detection, cybersecurity, and network monitoring. By identifying vulnerabilities before deployment, the proposed protocol can help motivate stronger defenses and more cautious use of clean benchmark performance.

A potential negative impact is the dual-use nature of score-directed cloaking, which may inform evasion attempts against deployed detectors. We mitigate this risk by presenting SGC as a controlled evaluation protocol, using bounded graph-edit budgets, and jointly studying AST as a normal-only defense. The work does not involve human-subject data, private datasets, or the release of high-risk models.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's scope as evaluating test-time graph cloaking for unsupervised graph-level anomaly detection, introducing score-directed graph cloaking, quantifying the directedness gap, and studying adversarial score training as a simple defense. These claims are supported by the formal problem setup, attack/defense methodology, and empirical results in Sections 2 and 3.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses limitations and future works in the conclusion.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theoretical theorems or proof-based guarantees.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sections 3 and Appendix C describe the detailed experimental setups for reproducibility.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The implementation code is included in the supplementary material.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies all the training and test details in Appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The main quantitative results are reported as a five-fold mean with standard deviation, and the tables explicitly state that the variability is computed across independent folds/runs. These error bars are reported for the experiments supporting the main claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C reports the detailed compute resources used for the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive impacts in improving the reliability of graph-level anomaly detectors under adaptive test-time manipulation. It also discusses the dual-use risk that score-directed cloaking could inform evasion attempts, and mitigates this risk by presenting the attack as an evaluation protocol together with a simple defense and reproducibility guidelines.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk pretrained models, scraped datasets, or human-subject data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original sources of the benchmark graph datasets and the evaluated baseline methods.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: the paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.