Attacking the Loop: Adversarial Attacks on Graph-Based Loop Closure Detection

Jonathan J. Y. Kim[1,3] [a], Martin Urschler[1,2] [b], Patricia J. Riddle[1] [c] and Jörg S. Wicker[1] [d]

[1]*School of Computer Science, University of Auckland, New Zealand*
[2]*Institute for Medical Informatics, Statistics and Documentation, Medical University Graz, Austria*
[3]*Callaghan Innovation, Auckland, New Zealand*

Keywords: Visual SLAM, Machine Learning, Adversarial Attacks, Graph Neural Networks, Loop Closure Detection.

Abstract: With the advancement in robotics, it is becoming increasingly common for large factories and warehouses to incorporate visual SLAM (vSLAM) enabled automated robots that operate closely next to humans. This makes any adversarial attacks on vSLAM components potentially detrimental to humans working alongside them. Loop Closure Detection (LCD) is a crucial component in vSLAM that minimizes the accumulation of drift in mapping, since even a small drift can accumulate into a significant drift over time. A prior work by Kim *et al.*, SymbioLCD2, unified visual features and semantic objects into a single graph structure for finding loop closure candidates. While this provided a performance improvement over visual feature-based LCD, it also created a single point of vulnerability for potential graph-based adversarial attacks. Unlike previously reported visual-patch based attacks, small graph perturbations are far more challenging to detect, making them a more significant threat. In this paper, we present *Adversarial-LCD*, a novel *black-box evasion* attack framework that employs an eigencentrality-based perturbation method and an SVM-RBF surrogate model with a Weisfeiler-Lehman feature extractor for attacking graph-based LCD. Our evaluation shows that the attack performance of *Adversarial-LCD* with the SVM-RBF surrogate model was superior to that of other machine learning surrogate algorithms, including SVM-linear, SVM-polynomial, and Bayesian classifier, demonstrating the effectiveness of our attack framework. Furthermore, we show that our eigencentrality-based perturbation method outperforms other algorithms, such as Random-walk and Shortest-path, highlighting the efficiency of *Adversarial-LCD*'s perturbation selection method.

1 INTRODUCTION

Simultaneous Localization and Mapping (SLAM) refers to a technique that involves creating a map of an unknown environment while simultaneously determining a device's location within the map. Visual SLAM (vSLAM) refers to a subset of SLAM being performed only using visual sensors (Mur-Artal and Tardos, 2016; Bescos et al., 2018; Schenk and Fraundorfer, 2019). With recent advancements in robotics, it is becoming increasingly common for large factories and warehouses to incorporate vSLAM-enabled automated robots into their operations. These robots are often designed to operate in close proximity with
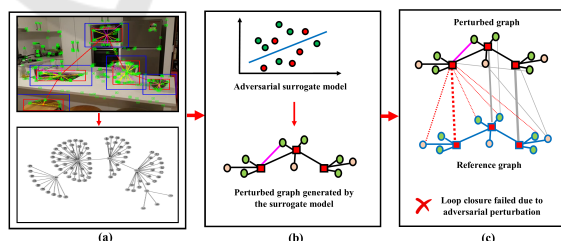

Figure 1: Basic overview of Adversarial-LCD (a) an input image is transformed into a unified graph structure (b) adversarial attacks are performed using a surrogate model (c) the target model is adversely affected by adversarial attacks.

human workers.

As a result, it is becoming essential to thoroughly examine vSLAM frameworks for potential vulnerabilities using various methods, such as adversarial attacks, to ensure the safety of human operators in such collaborative settings (Ikram et al., 2022).

[a] https://orcid.org/0000-0003-4287-4884
[b] https://orcid.org/0000-0001-5792-3971
[c] https://orcid.org/0000-0001-8616-0053
[d] https://orcid.org/0000-0003-0533-3368

Adversarial attacks (Dai et al., 2018; Wan et al., 2021) refer to a class of attacks that are designed to exploit vulnerabilities in Machine Learning (ML) models or systems by intentionally manipulating input data in a way that causes the model or system to produce incorrect results. A *white-box* attack (Zhang and Liang, 2019) is a type of adversarial attack that is carried out with complete knowledge of the target models' internal parameters and training data. In a *black-box* attack (Chen et al., 2022), the attacker has no knowledge of the target models' internal parameters or training data. They can only interact with the target model by querying input data and observing its output. An *evasion* attack (Wan et al., 2021) refers to the technique of creating an adversarial example by adding imperceptible perturbations to input data to adversely affect the target model.

Ikram *et al.* (Ikram et al., 2022) have shown that adversarial attacks can impact Loop Closure Detection (LCD) in vSLAM systems. They demonstrated this by modifying the environment, *i.e.* placing a simple high-textured patch in different places in the physical scene, thus adversely affecting the visual feature matching in LCD. However, since the attack requires putting visual patches in multiple places, it has potential to be discovered by human workers nearby.

The key challenge of vSLAM is to estimate the device trajectory accurately, even in the presence of feature matching inconsistencies and of *drift* from a sensor, as even small amounts of drift can accumulate to become potentially substantial by the end of the trajectory. To overcome the *drift* issue, vSLAM uses loop closure detection. It is a process of recognizing and correcting drift in the trajectory by revisiting previously visited locations, and it is essential for keeping consistent location and mapping within vSLAM.

SymbioLCD2 (Kim et al., 2022) simplified an image's semantic and spatial associations by developing a unified graph structure that integrates visual features, semantic objects, and their spatial relationship. While the development of a unified graph structure has improved performance over visual feature-based LCD (Mur-Artal and Tardos, 2016; Bescos et al., 2018), it has also introduced a potential point of vulnerability, where a malicious actor could carry out graph-based adversarial attacks to affect the LCD process adversely. Unlike visual patch-based attacks, graph-based attacks pose a higher threat, as small perturbations in a graph would be much harder to detect unless someone closely examines each graph instance.

In this paper, we study graph-based attacks by proposing our novel *black-box evasion* attack framework, called *Adversarial-LCD*, which employs an eigencentrality graph perturbation method, a Support Vector Machine (SVM) with Radial Basis Function (RBF) Kernel surrogate model, and a Weisfeiler-Lehman (WL) feature extractor. Firstly, it utilizes an eigencentrality perturbation method to select graph perturbations efficiently. This is accomplished by identifying the most well-connected nodes, which correspond to the most influential connections. Secondly, the WL feature extractor generates concatenated feature vectors from the perturbed graphs. This enables the surrogate model to learn directly from the graph-like search space. Thirdly, the framework incorporates an SVM-RBF surrogate model, which offers highly efficient performance even with a small number of training datasets. Figure 1 shows the basic framework and Figure 2 shows a detailed overview of our proposed *Adversarial-LCD*.

The main contributions of this paper are as follows:

- To the best of our knowledge, this is the first work presenting adversarial attacks on graph-based Loop Closure Detection.

- We propose *Adversarial-LCD*, a *black-box evasion* attack framework using an eigencentrality graph perturbation method and an SVM-RBF surrogate model with a WL feature extractor.

- We show that our *Adversarial-LCD* with the SVM-RBF surrogate model outperforms other ML surrogate algorithms, such as SVM-linear, SVM-polynomial and Bayesian classifier.

- We show that our *Adversarial-LCD* with eigencentrality graph perturbation method is more efficient than other perturbation methods, such as random-walk and shortest-path.

This paper is organized as follows. Section 2 reviews related work, Section 3 demonstrates our proposed method, Section 4 shows experiments and their results, and Section 5 concludes the paper.

2 RELATED WORK

This section reviews graph neural networks, graph perturbation methods, adversarial attacks on graph neural networks and adversarial attacks on loop closure detection.

Graph Neural Networks. Graph-structured data is a useful way of representing spatial relationships in a scene. Graph Neural Networks (GNNs) have become increasingly popular for efficiently learning relational representations in graph-structured data (Hamilton et al., 2017). GNN models are widely used for graph

classification (Zhang et al., 2018) and can generate graph embeddings in vector spaces to predict the similarity between a pair of graphs, making similarity reasoning more efficient (Li et al., 2019; Veličković et al., 2017).

In addition to GNNs, graph kernels (Siglidis et al., 2020) have emerged as a promising approach for graph classification. They enable kernelized learning algorithms, such as SVM and WL, to perform attributed subgraph matching and achieve state-of-the-art performance on graph classification tasks (Siglidis et al., 2020). The WL graph kernel utilizes a unique labelling scheme to extract subgraph patterns through multiple iterations. Each node's label is replaced with a label that consists of its original label and the subset of labels of its direct neighbours, which is then compressed to form a new label. The similarity between the two graphs is calculated as the inner product of their histogram vectors after the relabeling procedures (Shervashidze et al., 2011).

Graph Perturbation Methods. There are several graph perturbation methods that have been proposed for adversarial attacks on graph-based models (Dai et al., 2018; Wan et al., 2021). A graph perturbation aims to create a new graph that is similar to the original graph but with slight changes that can deceive the model into making incorrect predictions.

Random edge perturbation (Wan et al., 2021) involves randomly adding or removing edges from the graph to perturb the relationships between nodes. These perturbations in the graph increase the likelihood of false prediction on the target model. However, since the changes are made randomly, there is no guarantee that they will be effective in deceiving the model.

Shortest path perturbation (Kairanbay and Mat Jani, 2013) involves modifying the shortest path between two nodes in the graph by adding or removing edges to create a longer or shorter path. This can cause the model to make incorrect predictions by changing the relationships between nodes in the graph. This method is more targeted than random edge perturbation and has shown to be more effective in some cases (Kairanbay and Mat Jani, 2013).

Eigencentrality perturbation (Yan et al., 2014) involves modifying the centrality of nodes in the graph based on their eigencentrality, which is a measure of their importance in the network. This method targets the most important nodes in the graph and can have a significant impact on the model's predictions. Our graph perturbation method is based on eigencentrality.

Adversarial attacks on Graph Neural Networks. In recent years, GNNs have gained significant at-

tention and have been instrumental in various fields. However, like other deep neural networks, GNNs are also susceptible to malicious attacks. Adversarial attacks involve creating deceptive examples with minimal perturbations to the original data to reduce the performance of target models.

A *white-box* attack (Zhang and Liang, 2019) is an adversarial attack that is carried out with full knowledge of the internal parameters and training data of the target model. In a *black-box* attack (Chen et al., 2022), the attacker has no information about the target model's internal parameters or training data. They can only interact with the target model by querying input data and observing the model's output.

Poisoning attacks (Dai et al., 2018) involves deliberately injecting harmful samples during training to exploit the target machine learning model. Unlike data *poisoning*, *evasion* attacks (Zhang et al., 2021) generate adversarial perturbations to input data to adversely affect the target model.

In a *black-box evasion* attack (Wan et al., 2021) (Zhang et al., 2021), the attacker estimates the gradients of the target model's decision boundary with respect to the input data by repeatedly querying the target model with perturbed inputs and observing the corresponding outputs. Our method is a *black-box evasion* attack.

Adversarial Attacks on Loop Closure Detection. Adversarial attacks on LCD are still very nascent. Ikram *et al.* (Ikram et al., 2022) investigated the impact of adversarial attacks on LCD in vSLAM systems. They demonstrated that modifying the environment by placing a simple high-textured patch in various locations can negatively affect visual feature matching in LCD. However, to the best of our knowledge, there is no adversarial attack performed on a graph-based LCD. Graph-based attacks pose a higher threat, as unlike visible patches, small perturbations in a graph would be much harder to detect unless someone closely examines each graph instance.

3 PROPOSED METHOD

The overview of our proposed method is shown in Figure 2, and the reader may refer to (Kim et al., 2022) for further details on (a), (b) and (c). Our proposed methods are as follows - firstly, the original input graph from Figure 2 (c) is perturbed using the eigencentrality perturbation method. The target model is queried with the perturbed graphs, and observed attack losses are sent to the WL feature extraction process. Secondly, the WL feature extraction process extracts concatenated feature vectors
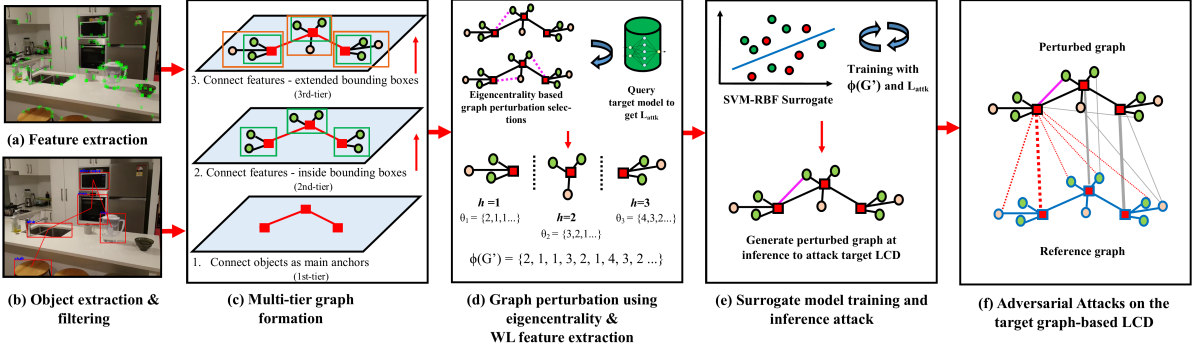
Figure 2: Detailed overview of *Adversarial-LCD*. (a) Visual feature extraction and generating a vBoW score. (b) Semantic object extraction from CNN and object filtering based on their location and size. (c) Multi-tier graph formation using semantic objects as main anchors - object and feature information gets transferred as node features, and distances between objects and features become edge features. (d) Graph perturbation selection using eigencentrality. The features from perturbed graphs are extracted using the WL feature extractor. (e) Training the surrogate model and generating perturbed graphs for adversarial attacks on the target model. The purple edge indicates an example perturbation. (f) Adversarial attacks are performed on the target graph-based LCD, causing it to perform incorrect loop closure.

from the set of perturbed graphs. This approach enables the surrogate model to learn on the graph-like search space efficiently. Lastly, the SVM-RBF surrogate model is trained using the extracted WL features of perturbed graphs and their corresponding attack losses. At inference time, we use the surrogate model to attack the target graph-based LCD to degrade its performance.

3.1 Framework

We propose *Adversarial-LCD* as the framework for incorporating our adversarial attacks into the LCD framework. The *Adversarial-LCD* framework was created using SymbioLCD2 (Kim et al., 2022), which uses a unified graph structure as an input to its Weisfeiler-Lehman subgraph matching algorithm to predict loop closure candidates.

The *Adversarial-LCD* is made up of three modules. The first module, which we call the *graph-module*, does the visual and semantic object extraction to create a multi-tier graph, as shown in Figure 2(a - c). The third module, which we call the *target-LCD*, contains the loop closure detection model shown in Figure 2(f). The second module, which we call the *attack-module*, shown in Figure 2(d & e), is situated between the first and third modules. The *attack-module* has access to the input graph G as it is being sent from the *graph-module* to the *target-LCD*.

3.2 Problem Setup

We perform *black-box evasion* attacks, where our *attack-module* has no access to the training data or pa-

rameters on the *target-LCD* model f_t. However, it has access to the input graph G from the *graph-module*, and it can query the *target-LCD* with a perturbed input graph G' and observe the *target-LCD* model output $f_t(G')$.

Our adversarial attack aims to degrade the predictive performance of the *target-LCD* via a *black-box* maximization,

$$max_{G'} \; \mathcal{L}_{attk}(f_t(G'),y), \quad (1)$$

where $\mathcal{L}_{attk}$ is the attack loss function, G' is the perturbed version of G, and y refers to the correct label of the original graph G.

3.3 Perturbation Selections Using Eigencentrality

Changing a large number of graph connections can be expensive and runs the chance of being detected if the perturbation amount becomes excessive. Therefore, we use eigencentrality for effectively selecting the perturbations within the perturbation budget of $\beta = rn^2$, where r refers to *perturbation ratio* (Chen et al., 2022), and n refers to the number of nodes. Eigencentrality can identify the most well-connected nodes, i.e., the most influential connections, which have a higher chance of disruption when the connections are added or subtracted.

Given the input graph $G = (V,E)$ with v vertices and its neighbouring vertices u, an adjacency matrix A could be defined as a set of (a_v, u), where $a_v = 1$ if the vertex v is connected to the vertex u, or $a_v = 0$ if it is not connected to the vertex u. The eigencentrality score X for a vertex v can be defined as,

$$X_v = \frac{1}{\lambda} \sum_{u \varepsilon V} AX_u, \quad (2)$$

or in vector notation,

$$Ax = \lambda x, \qquad (3)$$

where λ is a constant. The eigencentrality amplifies the components of the vector corresponding to the largest eigenvalues, i.e. *centrality*. We use the eigencentrality to iteratively generate a set of perturbed graphs and query the *target-LCD* model $f_t(G')$ until the attack is successful, or until the maximum query budget is exhausted. The resulting perturbed graphs, the original graph and their attack losses are sent to the Weisfeiler-Lehman feature extraction process.

3.4 Weisfeiler-Lehman Feature Extraction

Building on the work of Wan *et al.* (Wan et al., 2021), we leverage the WL feature extractor to extract concatenated feature vectors from the set of perturbed graphs generated in the previous step. This approach enables us to directly train a surrogate model on the graph-like search space in an efficient manner. Given the initial node feature $x_0(v)$ of node v, WL feature extractor iteratively aggregates features of v with features of its neighbour u,

$$x_0^h(v) = aggregate(x^h(v), x^h(u_1), ..., x^h(u_i)), \qquad (4)$$

where h refers to the number of iterations. At each h, feature vector $\phi_h(G')$ can be defined as,

$$\phi_h(G') = (x_0^h(v), x_1^h(v), ..., x_i^h(v)). \qquad (5)$$

The final feature vector at the end of the total number of iterations H,

$$\phi(G') = concat(\phi_1(G'), \phi_2(G'), ..., \phi_H(G')), \qquad (6)$$

is sent to the surrogate model for training.

3.5 Surrogate Model

The SVM is a widely recognized ML algorithm for its simplicity and effectiveness in finding the optimal hyperplane. It also offers kernel functions, which are a potent tool for navigating high-dimensional spaces. With kernel functions, SVM can directly map the data into higher dimensions without the need to transform the entire dataset (Han and Scarlett, 2022). Thus, we utilize SVM-RBF as our surrogate model as it can deliver efficient training performance with Gaussian probabilistic output in a binary classification setting. We train our SVM-RBF surrogate with WL feature vectors $\phi(G')$ and their attack losses $y' = \mathcal{L}_{attk}$ as inputs.

RBF combines various polynomial kernels with differing degrees to map the non-linear data into a higher dimensional space, so that it can be separated using a hyperplane. RBF kernel maps the data into a higher-dimensional space by,

$$K(\phi(G_i'), \phi(G_j')) = \exp(-\frac{||\phi(G_i') - \phi(G_j')||^2}{2\sigma^2}), \qquad (7)$$

where σ is a tuning parameter, based on the standard deviation of a dataset. To simplify, we assume $\gamma = \frac{1}{2\sigma^2}$, which leads to,

$$K(\phi(G_i'), \phi(G_j')) = \exp(-\gamma||\phi(G_i') - \phi(G_j')||^2). \qquad (8)$$

With the kernel function, the optimization of the SVM surrogate model can be written as,

$$min_\alpha \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i' y_j' K(\phi(G_i'), \phi(G_j')) - \sum_{i=1}^{N} \alpha_i, \qquad (9)$$

$$s.t. \sum_{i=1}^{N} \alpha_i y_i' = 0 \ \& \ \alpha_i \geq 0, \qquad (10)$$

where α refers to a Lagrange multiplier (Rockafellar, 1993) corresponding to the training data $\phi(G')$.

4 EXPERIMENTS

We have evaluated *Adversarial-LCD* with the following experiments. Section 4.1 shows datasets and evaluation parameters used in the experiments. Section 4.2 evaluates *Adversarial-LCD* with the SVM-RBF surrogate model against other machine learning surrogate models. Section 4.3 evaluates the eigencentrality perturbation method against other perturbation algorithms.

4.1 Setup

For evaluating our *Adversarial-LCD*, we have selected five publicly available datasets with multiple objects and varying camera trajectories. We selected fr2-desk and fr3-longoffice from the TUM dataset (Sturm et al., 2012), and uoa-lounge, uoa-kitchen and uoa-garden from the University of Auckland multi-objects dataset (Kim et al., 2021). The details of the datasets are shown in Table 1b. Table 1a shows evaluation parameters and Figure 3 shows examples from each dataset. All experiments were performed on a PC with Intel i9-10885 and Nvidia GTX2080.

4.2 Evaluation Against Other Machine Learning Surrogate Models

We conducted a benchmark of *Adversarial-LCD* with the SVM-RBF surrogate model against three
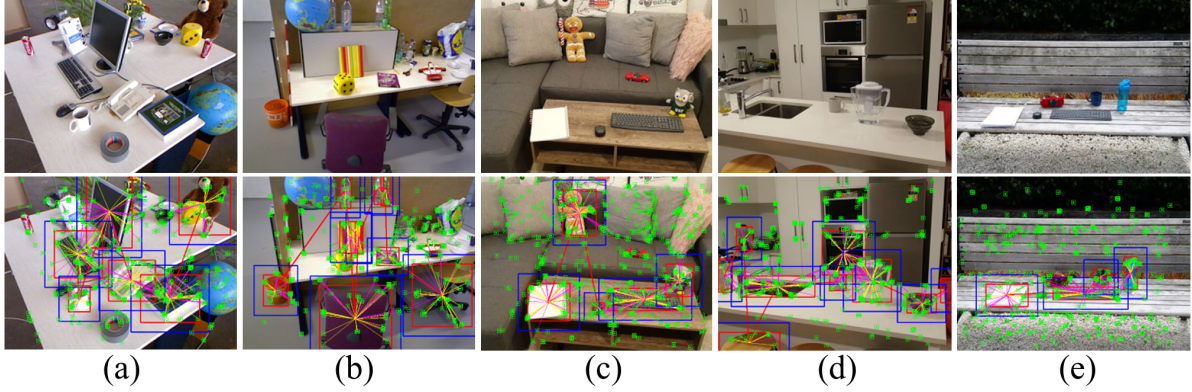
Figure 3: Evaluation datasets. (a) fr2-desk (b) fr3-longoffice (c) uoa-lounge (d) uoa-kitchen (e) uoa-garden.

Table 1: Parameters and Datasets.

Parameters	Value
Epochs	200
Rand. state	42
r	$3e^{-4}$
λ	0.1
γ	$\frac{1}{2\sigma^2}$
α	0.05

(a) Parameters.

Dataset	Source	No. of frames	Image Res.
fr2-desk	TUM	2965	640x480
fr3-long.	TUM	2585	640x480
lounge	ours	1841	640x480
kitchen	ours	1998	640x480
garden	ours	2148	640x480

(b) Datasets descriptions.

other ML surrogate models, SVM-linear, SVM-polynomial, and Bayesian classifier. To evaluate each surrogate model, we attacked the *target-LCD* model and recorded the decline in its accuracy. To simulate a realistic scenario where a large number of changes to the graph connections would easily raise suspicion, we allowed only a small perturbation budget of $r = 3e^{-4}$ for the experiment. To account for the non-deterministic nature of the algorithms, we performed the evaluation ten times. The results, presented in Table 2, indicate that on average, *Adversarial-LCD* with SVM-RBF achieved the highest decline in accuracy compared to the other algorithms, surpassing SVM-linear by 12.6%, SVM-polynomial by 7.3%, and Bayesian classifier by 2.7%.

To assess the statistical robustness of our findings, we utilized Autorank (Herbold, 2020) to further analyze the performance of each algorithm. Autorank is an automated ranking algorithm that fol-

lows the guidelines proposed by Demšar (Demsar, 2006) and employs independent paired samples to determine the differences in central tendency, such as median (MED), mean rank (MR), and median absolute deviation (MAD), for ranking each algorithm. It also provides the critical difference (CD), which is a statistical technique utilized to ascertain whether the performance difference between two or more algorithms is statistically significant. For the Autorank evaluation, we used an $\alpha = 0.05$. The result presented in Table 3 shows that *Adversarial-LCD* received the highest ranking against the other ML algorithms, and Figure 4 shows that there is a critical difference between *Adversarial-LCD* and the other algorithms, highlighting the effectiveness of our SVM-RBF surrogate model.

Table 2: The decline in LCD accuracy using different surrogate models

Dataset	SVM-linear	SVM-Poly	Bayesian	Adv-LCD
fr2-desk	-13.29	-17.37	-22.22	**-27.92**
fr3-longoffice	-14.10	-19.72	-26.94	**-29.33**
lounge	-12.89	-19.89	-26.21	**-28.72**
kitchen	-15.67	-18.64	**-24.99**	-24.66
garden	-13.22	-17.89	-18.46	**-21.69**
Average	-13.83	-18.50	-23.76	**-26.46**
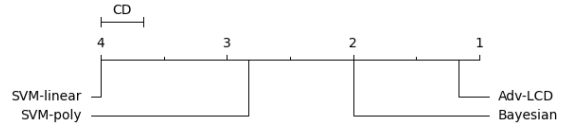
Figure 4: Critical Difference diagram.

Table 3: Autorank analysis on different surrogate models.

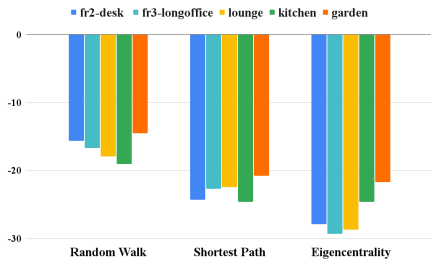|	MR	MED	MAD	CI	γ	Mag.
Adv-LCD	3.83	-27.19	1.83	[-27.9, -26.4]	0.0	neg.
Bayesian	3.00	-24.37	1.99	[-24.9, -23.7]	-0.9	large
SVM-poly	2.16	-19.41	0.48	[-19.7, -19.1]	-3.9	large
SVM-linear	1.00	-13.56	0.44	[-13.8, -13.2]	-6.8	large

Figure 5: The decline in LCD accuracy using different perturbation methods.

4.3 Ablation Study: The Effectiveness of Eigencentrality Against Other Perturbation Methods

We conducted two evaluations to assess the effectiveness of the eigencentrality perturbation method. To account for the non-deterministic nature of the algorithms, we performed both evaluations ten times. For the first evaluation, we kept all parameters, including the perturbation budget, identical to the previous evaluation. The results presented in Table 4 and Figure 5 show that, on average, the eigencentrality method outperforms Random-walk by 9.6% and Shortest-path by 4.0%. The result demonstrates that our perturbation method based on node centrality is more effective than modifying shortest paths or selecting perturbations randomly.

For the second evaluation, we constrained the perturbation budgets further to compare the perturbation-efficiency of the eigencentrality method against other methods. The evaluation was performed using the *fr2-desk* dataset. The result presented in Table 5 shows that the eigencentrality method outperformed both Random-walk and Shortest-path across all evaluated perturbation budgets. On average, the eigencentrality method surpassed Random-walk by 7.9% and Shortest-path by 3.1%, highlighting the strong perturbation-efficiency of our method.

Table 4: The decline in LCD accuracy using different perturbation methods.

Dataset	Random Walk	Shortest Path	Eigencentrality
fr2-desk	-15.66	-24.30	**-27.92**
fr3-longoffice	-16.72	-22.69	**-29.33**
lounge	-17.95	-22.46	**-28.72**
kitchen	-19.04	-24.63	**-24.66**
garden	-14.55	-20.76	**-21.69**
Average	-16.548	-22.36	**-26.46**

5 CONCLUSION AND FUTURE WORK

In this paper, we presented *Adversarial-LCD*, a novel *black-box evasion* attack framework, which uses an eigencentrality graph perturbation method and an SVM-RBF surrogate model with a Weisfeiler-Lehman feature extractor. We showed that our *Adversarial-LCD* with SVM-RBF surrogate model outperformed other ML surrogate algorithms, such as SVM-linear, SVM-polynomial and Bayesian classifier, demonstrating the effectiveness of *Adversarial-LCD* framework.

Table 5: The decline in LCD accuracy at different perturbation budgets (fr2-desk).

Budget	Random Walk	Shortest Path	Eigencentrality
$r = 1e^{-4}$	-1.5	-3.45	**-5.02**
$r = 2e^{-4}$	-7.33	-11.22	**-16.33**
$r = 3e^{-4}$	-15.66	-24.30	**-27.92**
Average	-8.16	-12.95	**-16.09**

Furthermore, we demonstrated that our perturbation method based on eigencentrality outperformed other algorithms such as Random-walk and Shortest-path in generating successful adversarial perturbations, highlighting that perturbing nodes based on their centrality is more efficient than randomly selecting perturbations or modifying the shortest paths between nodes.

Our future research will focus on exploring adversarial defence techniques, such as adversarial learning, for graph-based loop closure detection.

ACKNOWLEDGEMENTS

REFERENCES

Bescos, B., Facil, J., Civera, J., and Neira, J. (2018). DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes. *IEEE Robotics and Automation Letters*, 3:4076 – 4083.

Chen, J., Zhang, D., Ming, Z., Huang, K., Jiang, W., and Cui, C. (2022). GraphAttacker: A General Multi-Task Graph Attack Framework. *IEEE Transactions on Network Science and Engineering*, 9(2):577–595.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. (2018). Adversarial Attack on Graph Structured Data. In *International Conference on Machine Learning*, volume 80, pages 1115–1124.

Demsar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7:1–30.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, pages 1025 – 1035.

Han, E. and Scarlett, J. (2022). Adversarial Attacks on Gaussian Process Bandits. In *International Conference on Machine Learning*, volume 162, pages 8304–8329.

Herbold, S. (2020). Autorank: A Python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48):2173.

Ikram, M. H., Khaliq, S., Anjum, M. L., and Hussain, W. (2022). Perceptual Aliasing++: Adversarial Attack for Visual SLAM Front-End and Back-End. *IEEE Robotics and Automation Letters*, 7(2):4670–4677.

Kairanbay, M. and Mat Jani, H. (2013). A Review and Evaluations of Shortest Path Algorithms. *International Journal of Scientific & Technology Research*, 2:99–104.

Kim, J. J. Y., Urschler, M., Riddle, P., and Wicker, J. (2021). SymbioLCD: Ensemble-Based Loop Closure Detection using CNN-Extracted Objects and Visual Bag-of-Words. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5425–5432.

Kim, J. J. Y., Urschler, M., Riddle, P., and Wicker, J. (2022). Closing the Loop: Graph Networks to Unify Semantic Objects and Visual Features for Multi-object Scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4352–4358.

Li, Y., Gu, C., Dullien, T., Vinyals, O., and Kohli, P. (2019). Graph Matching Networks for Learning the Similarity of Graph Structured Objects. In *International Conference on Machine Learning*, pages 3835–3845.

Mur-Artal, R. and Tardos, J. (2016). ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics*, 33:1255–1262.

Rockafellar, R. T. (1993). Lagrange Multipliers and Optimality. *Society for Industrial and Applied Mathematics Review*, 35(2):183–238.

Schenk, F. and Fraundorfer, F. (2019). RESLAM: A real-time robust edge-based SLAM system. In *International Conference on Robotics and Automation*, pages 154–160.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-Lehman Graph Kernels. *Journal of Machine Learning Research*, 12(77):2539–2561.

Siglidis, G., Nikolentzos, G., Limnios, S., Giatsidis, C., Skianis, K., and Vazirgiannis, M. (2020). GraKeL: A Graph Kernel Library in Python. *Journal of Machine Learning Research*, 21(54):1–5.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A Benchmark for the Evaluation of RGB-D SLAM Systems. In *International Conference on Intelligent Robot Systems*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2017). Graph Attention Networks. *International Conference on Learning Representations*.

Wan, X., Kenlay, H., Ru, B., Blaas, A., Osborne, M., and Dong, X. (2021). Adversarial Attacks on Graph Classification via Bayesian Optimisation. *Advances in Neural Information Processing Systems*, 34.

Yan, X., Wu, Y., Li, X., Li, C., and Hu, Y. (2014). Eigenvector perturbations of complex networks. *Statistical Mechanics and its Applications*, 408:106–118.

Zhang, H., Wu, B., Yang, X., Zhou, C., Wang, S., Yuan, X., and Pan, S. (2021). Projective Ranking: A Transferable Evasion Attack Method on Graph Neural Networks. In *International Conference on Information and Knowledge Management*, pages 3617–3621.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. (2018). An End-to-End Deep Learning Architecture for Graph Classification. In *Association for the Advancement of Artificial Intelligence*, pages 4438–4445.

Zhang, Y. and Liang, P. (2019). Defending against White-box Adversarial Attacks via Randomized Discretization. In *International Conference on Artificial Intelligence and Statistics*, volume 89, pages 684–693.