Relaxing Graph Transformers for Adversarial Attacks

Philipp Foth [1] **Lukas Gosch** [1 2] **Simon Geisler** [1] **Leo Schwinn** [1] **Stephan Günnemann** [1]

Abstract

Existing studies have shown that Graph Neural Networks (GNNs) are vulnerable to adversarial attacks. Even though Graph Transformers (GTs) surpassed Message-Passing GNNs on several benchmarks, their adversarial robustness properties are unexplored. However, attacking GTs is challenging due to their Positional Encodings (PEs) and special attention mechanisms which can be difficult to differentiate. We overcome these challenges targeting three representative architectures based on (1) random-walk PEs, (2) pair-wise-shortest-path PEs, and (3) spectral PEs – and propose the first adaptive attacks for GTs. We leverage our attacks to evaluate robustness to (a) structure perturbations on node classification; and (b) node injection attacks for (fake-news) graph classification. Our evaluation reveals that they can be catastrophically fragile and underlines our work's importance and the necessity for adaptive attacks.

1. Introduction

Graphs are versatile data structures that have applications in a wide range of different domains and Graph Neural Networks (GNNs) have become the tool of choice for many learning tasks on graphs. Given the increasing popularity of GNNs, multiple studies in the past years have developed adversarial attacks for GNNs and analyzed their robustness [1]–[3]. These studies mostly focus on Message-Passing GNNs (MPNNs), such as the Graph Convolutional Network (GCN) [4] and show that GNNs are vulnerable to even slight graph structure perturbations [1].

Recently, Graph Transformers (GTs) have started to increasingly outperform MPNNs in many common benchmarks [5]. They address inherent limitations of MPNNs such as *over-smoothing*, *over-squashing*, and limited receptive fields [5]. Despite their increasing popularity, the adversarial robustness of GTs is entirely unexplored. Consequently, the unknown stability of GTs poses a substantial risk in practical applications, where robustness is crucial. Yet evaluating, their robustness is non-trivial since they employ modified attention mechanisms and Positional Encodings (PEs) that are often non-differentiable w.r.t. the graph structure. This renders the immediate application of standard gradient-based adaptive attacks impossible. In prior work, similar issues were addressed with adaptive attacks, tailored to specific architectures [6]–[8] including GNNs [9].

To provide the first analysis on the robustness of GTs, we propose continuous relaxations and perturbation approximations that enable us to apply adaptive attacks to three representative GT architectures. **1)** Graph Inductive bias Transformer (**GRIT**) [10], based on random walk PEs; **2) Graphormer** [11], which uses pairwise shortest path PEs; and **3)** Spectral Attention Network (**SAN**) [12], with spectral PEs. We evaluate the adversarial robustness of the three GT models for node and graph classification. To this end, we propose a node injection attack (NIA) which we apply to attack fake news detection. Fig. 1 provides a direct comparison between the robustness of different GNNs and highlights crucial robustness vulnerabilities of GTs.

2. Background

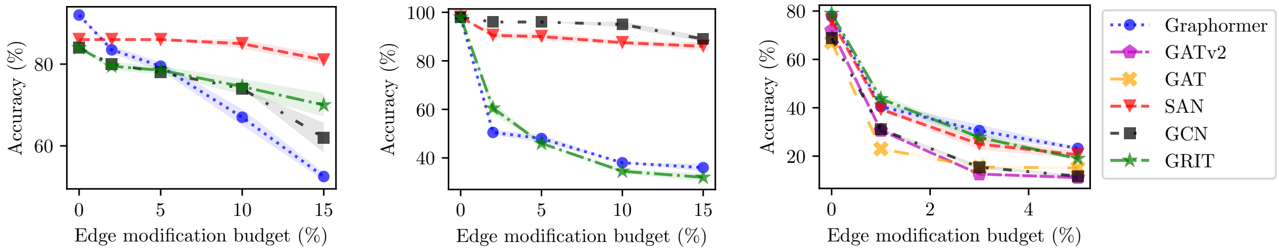
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected attributed graph with n nodes $\mathcal{V} = \{v_1, ..., v_n\}$ and m edges. Let $\boldsymbol{x}_i \in \mathbb{R}^d$ be the feature vector of node v_i. Then the graph can be defined as $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ with its symmetric binary adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$ and node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$. The diagonal degree matrix $\boldsymbol{D}$ with entries $\boldsymbol{D}_{ii} = \mathsf{deg}(v_i) = \sum_{j=1}^{n} \boldsymbol{A}_{ij}$ and the normalized symmetric graph Laplacian matrix $\boldsymbol{L}_{sym} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ can both be derived from $\boldsymbol{A}$. The GNNs considered in this work are functions $f_\theta(\boldsymbol{A}, \boldsymbol{X})$ with model parameters θ, and $\boldsymbol{H}^{(l)}$ denotes the updated node representations in the model layers.

Structure attacks. The attack objective is described by the following optimization problem:

$$\max_{\tilde{\boldsymbol{A}} \text{ s.t. } \|\tilde{\boldsymbol{A}} - \boldsymbol{A}\|_0 < \Delta} \mathcal{L}_{atk}(f_\theta(\tilde{\boldsymbol{A}}, \boldsymbol{X})) \tag{1}$$

[1] Department of Computer Science & Munich Data Science Institute, Technical University of Munich, Germany [2] Munich Center for Machine Learning (MCML). Correspondence to: Philipp Foth <p.foth@tum.de>, Lukas Gosch <l.gosch@tum.de>.

Published at the 2^{nd} Differentiable Almost Everything Workshop at the 41^{st} International Conference on Machine Learning, Vienna, Austria. July 2024. Copyright 2024 by the author(s).



(a) UPFD politifact.

(b) UPFD gossipcop.

(c) CLUSTER.

Figure 1. The classification accuracy for different GNNs with varying attack budget on the two UPFD Twitter fake news datasets (graph classification, node injection attacks) and CLUSTER (node classification, structure attack). The strongest attack for each budget is shown.

where f_θ is the GNN model with fixed parameters θ, $\mathcal{L}_{atk}$ is a suitable attack loss function, and $\tilde{\boldsymbol{A}} \in \{0,1\}^{n\times n}$ is the discrete perturbed adjacency matrix in relation to $\boldsymbol{A}$ with the number of edge flips bounded by the budget Δ. It is convenient to model the perturbation as a function of the binary matrix indicating the edge flips $\boldsymbol{B} \in \{0,1\}^{n\times n}$:

$$\tilde{\boldsymbol{A}} = \boldsymbol{A} + \delta\boldsymbol{A}, \qquad \delta\boldsymbol{A} = (\boldsymbol{1}_n\boldsymbol{1}_n^\top - 2\boldsymbol{A}) \odot \boldsymbol{B} \qquad (2)$$

with element-wise product $\odot$. Applying a continuous relaxation $\boldsymbol{B} \in [0,1]^{n\times n}$ to the combinatorial problem makes optimization with gradient descent possible. In this setting, the entry $\boldsymbol{B}_{ij}$ represents the probability that the edge (v_i, v_j) is flipped. Discrete perturbations can then be sampled from the continuous solution. The budget constraint becomes $\mathbb{E}[\mathsf{Bernoulli}(\boldsymbol{B})] = \sum \boldsymbol{B}_{ij} \leq \Delta$, which can be dealt with by using projected gradient descent [13]. For large graphs, optimizing over all entries in $\boldsymbol{B}$ at once becomes infeasible. Projected Randomized Block Coordinate Descent (PRBCD) solves this with a strategy that optimizes over sampled random blocks of limited size [14].

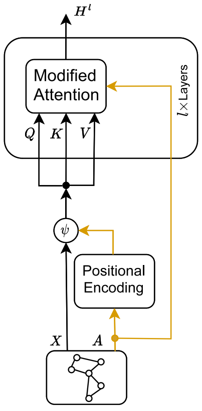


Figure 2. Generic GT architecture.

Graph transformers. GTs apply the popular transformer architecture for sequences [15] to arbitrary graphs. In this work, we focus on GTs that apply global self-attention, where each node can attend to all other nodes. A ‘vanilla’ structure-unaware self-attention head is defined as:

$$\mathsf{Attn}(\boldsymbol{H}) = \sigma\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V} \qquad (3)$$

where $\boldsymbol{Q} = \boldsymbol{H}\boldsymbol{W}_q, \boldsymbol{K} = \boldsymbol{H}\boldsymbol{W}_k \in \mathbb{R}^{n\times d_k}$ are the *query* and *key* projections, $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_v \in \mathbb{R}^{n\times d}$ is the *value* projection, and σ is the softmax function. A generic GT is depicted in Fig. 2. The individual attention scores are:

$$w_{ij} = \boldsymbol{W}_q\boldsymbol{h}_i \cdot \boldsymbol{W}_k\boldsymbol{h}_j/\sqrt{d},$$
$$\alpha_{ij} = \sigma(w_{ij}) = \frac{e^{w_{ij}}}{\sum_k e^{w_{ik}}} \qquad (4)$$

For structure awareness, prior works commonly add Positional Encodings (PEs) to the node features: $\boldsymbol{H}^{(0)} = \boldsymbol{X} + \psi(\boldsymbol{A})$. We categorize the PEs roughly in three main categories: (1) random walk encodings, (2) distance encodings, and (3) spectral encodings. In addition to PEs, many GTs adapt the attention mechanism to capture relevant topological aspects of the graph explicitly in the attention operation.

3. Graph transformer relaxations

The main obstacles for gradient-based structure attacks on GTs f_θ are PEs and attention mechanisms that are designed for a discrete graph structure. As a result, the model output is often a discontinuous function of the continuous input adjacency matrix $\tilde{\boldsymbol{A}} \in [0,1]^{n\times n}$, rendering continuous optimization ineffective. We identify such components in the GT models and continuously relax them: $\tilde{f}_\theta$. For designing the continuous relaxations, we identify three main principles:

Principle I: Relaxed and target models coincide for discrete inputs. The prediction should equal $\tilde{f}_\theta(\boldsymbol{A}) = f_\theta(\boldsymbol{A})$ for any discrete adjacency $\boldsymbol{A} \in \{0,1\}^{n\times n}$.

Principle II: $\tilde{f}_\theta$ can interpolate “smoothly” between any different discrete graphs. In other words, $\tilde{f}_\theta(\tilde{\boldsymbol{A}})$ should be continuous w.r.t. $\tilde{\boldsymbol{A}}$ but does not need to be continuously differentiable.

Principle III: The relaxed model $\tilde{f}_\theta$ must be efficient. It is a critical property that the relaxation does not excessively increase memory and runtime complexity.

Next, we introduce a representative GT for each of the PE categories and propose continuous relaxations for problematic components based on these principles.

Random-walk-based GRIT. The components of the GRIT model [10] are already continuous and do not require additional relaxations for gradient-based attacks. Details on the architecture are given in Appendix B.

Distance-based Graphormer. The Graphormer model [11] learns a PE embedding vector $\boldsymbol{z} \in \mathbb{R}^d$ for each integer degree value (# of neighbors) which are added to the node features according to their degrees: $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i + \boldsymbol{z}_{\mathsf{deg}(v_i)}$. To allow for continuous degrees, we linearly interpolate between the PE embeddings of the two closest integer values:

$$\begin{aligned} \tilde{\boldsymbol{z}}_{\mathsf{deg}(v_i)} &= \eta \cdot \boldsymbol{z}_{\mathsf{d}_l+1} + (1-\eta) \cdot \boldsymbol{z}_{\mathsf{d}_l} \\ \text{with} \quad \mathsf{d}_l &= \lfloor \mathsf{deg}(v_i) \rfloor, \quad \eta = \mathsf{deg}(v_i) - \mathsf{d}_l \end{aligned} \tag{5}$$

Similarly, a learnable scalar $b \in \mathbb{R}$ is assigned to each discrete Shortest Path Distance (SPD). This value is added as a bias to the raw attention scores: $\hat{w_{ij}} = w_{ij} + b_{\mathsf{spd}(v_i,v_j)},.$ When a very small edge probability lies on a (simple) shortest path, the path is less likely to exist in the discrete sampled adjacency matrix. Therefore, low edge probabilities should only marginally affect the original SPDs. To model this relationship, we use the reciprocal of the continuous adjacency matrix $\boldsymbol{R}_{ij} = 1/\tilde{\boldsymbol{A}}_{ij}$ for the SPDs. We interpolate again to handle the continuous distance values and compute:

$$\begin{aligned} \tilde{b}_{\mathsf{spd}(v_i,v_j)} &= \eta \cdot b_{\mathsf{s}_l+1} + (1-\eta) \cdot b_{\mathsf{s}_l} \\ \text{with} \quad \mathsf{s}_l &= \lfloor \mathsf{rspd}_{ij} \rfloor, \quad \eta = \mathsf{rspd}_{ij} - \mathsf{s}_l \end{aligned} \tag{6}$$

where $\mathsf{rspd}_{ij} = \mathsf{spd}(v_i, v_j | \boldsymbol{R})$ is the sum of edge weights that lie on the shortest path from v_i to v_j for the graph defined by $\boldsymbol{R}$. Similar to ReLU activations, the distances are not differentiable everywhere. However, we use the sum over one possible shortest path as a proxy. Backpropagation through this sum yields the gradients w.r.t the continuous adjacency matrix. Note that for discrete edge probability values 0 and 1, the reciprocal edge weights become $-\infty$ and 1, respectively, yielding the original distances. Hence, we do not alter the clean predictions if $\delta\boldsymbol{A} = \boldsymbol{B} = \boldsymbol{0}$ (see Principle I).

Spectral SAN. The SAN architecture [12] uses learned Laplacian-based PEs (LPEs), starting with the eigen-decomposition of the Laplacian $\boldsymbol{L}_{sym} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, where diagonal entries of $\boldsymbol{\Lambda}_{ii} = \lambda_i$ are the eigenvalues of $\boldsymbol{L}_{sym}$ in ascending order, and the columns of $\boldsymbol{U}$ are the corresponding eigenvectors. While the Laplacian matrix itself is a continuous functions of the entries in the adjacency matrix, its eigen-decomposition poses challenges for gradient computation, especially w.r.t the eigenvectors. To avoid direct gradient computation, we use results from matrix perturbation theory [16], [17] to approximate the perturbed eigen-decomposition as a simpler function of the input perturbation. The perturbation is $\delta\boldsymbol{L}_{sym} = \tilde{\boldsymbol{L}}_{sym} - \boldsymbol{L}_{sym}$, where $\tilde{\boldsymbol{L}}_{sym}$ is the Laplacian of the continuous adjacency matrix $\tilde{\boldsymbol{A}}$. The first-order approximations for the eigenvalues and vectors are [17]:

$$\tilde{\boldsymbol{\Lambda}} = \boldsymbol{\Lambda} + \delta\boldsymbol{\Lambda}, \qquad \delta\boldsymbol{\Lambda} \approx \mathsf{diag}(\boldsymbol{U}^\top \delta\boldsymbol{L}\,\boldsymbol{U}) \tag{7}$$

$$\tilde{\boldsymbol{U}} = \boldsymbol{U} + \delta\boldsymbol{U}, \quad \delta\boldsymbol{U} \approx -\boldsymbol{U}\ \left(\boldsymbol{\Pi} \odot \boldsymbol{U}^\top \delta\boldsymbol{L}\,\boldsymbol{U}\right),$$

$$\text{where} \quad \boldsymbol{\Pi}_{ij} = \begin{cases} \frac{1}{\lambda_i - \lambda_j} & \text{if } \lambda_i \neq \lambda_j \\ 0 & \text{else} \end{cases} \tag{8}$$

For the approximation to hold with repeated eigenvalues, special care for the choice of arbitrary eigenvectors that span the eigenspace is required, as described by Bamieh [17].

The SAN attention mechanism computes separate attention scores between connected and unconnected nodes, using different key and query weights. A hyperparameter $\gamma \in \mathbb{R}^+$ controls how the two scores are relatively scaled, varying the bias towards sparse or full attention. To relax the binary decision of whether an edge is real or fake, we add edges with probability between 0 and 1 to both attention mechanisms. The contribution of the uncertain edges is weighted relative to the probabilities of being real or fake using a corresponding bias:

$$\begin{aligned} \tilde{\alpha}_{ij} \ &= \frac{\gamma}{1+\gamma}\sigma\left(w_{\mathsf{fake},ij} + \log(1 - \tilde{\boldsymbol{A}}_{ij})\right) \\ &\quad + \frac{1}{1+\gamma}\sigma\left(w_{\mathsf{real},ij} + \log(\tilde{\boldsymbol{A}}_{ij})\right) \end{aligned} \tag{9}$$

For a discrete real edge, i.e., $\tilde{\boldsymbol{A}}_{ij} = 1$, the logarithm terms become $-\infty$ and 0 respectively. The equation then simplifies to $\alpha_{ij} = \frac{1}{1+\gamma}\mathsf{softmax}\left(w_{\mathsf{real},ij}\right)$, thus only contributing to the 'real' attention scores, as in the original SAN. The same can be shown for fake edges.

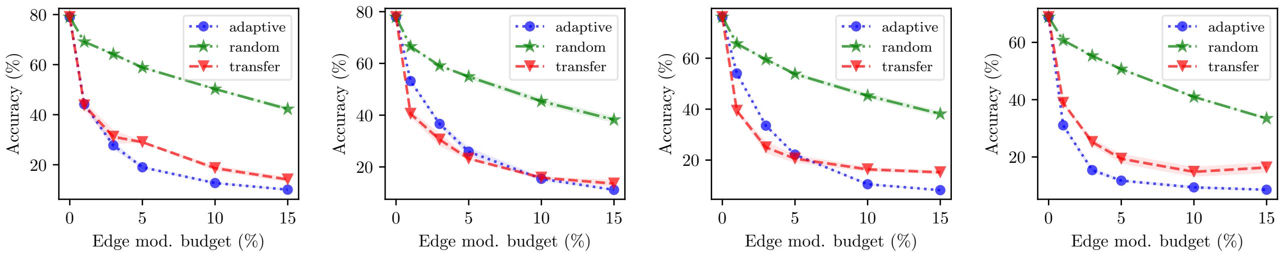
4. Node injection attack

We also consider the relevant case of inserting nodes into an existing graph structure. In contrast to the usual framing of Node Injection Attack (NIA), where the attacker also chooses the node features for the new *vicious* nodes [18], we connect existing nodes from other graphs of an inductive graph dataset. Therefore, the nodes' features are fixed but physically realizable even if, e.g., they represent embeddings of natural language. Details are shown in Appendix C.

During the structure attack optimization, probabilities are assigned to *edges*, yet for NIAs, *node* probabilities are also relevant to allow for a smooth node insertion. To obtain these from the edges, we propose a simple iterative computation. For a continuous connected graph, let $\mathcal{N}(v_i)$ be the 1-hop neighborhood of node v_i. We can approximate the probability p_i of v_i being connected to the graph, by using the probability that it is connected to its neighbors and the probability that these neighbors are connected to the graph. We start with the assumption that all nodes are connected to the graph: $p_i^{(0)} = 1$, and update using the edge probabilities:

$$p_i^{(t+1)} = 1 - \prod_{v_j \in \mathcal{N}(v_i)} (1 - \boldsymbol{A}_{ij} \cdot p_j^{(t)}) \tag{10}$$

To to ensure that the model output is continuous w.r.t. node injections, for graph level tasks we weigh the nodes' contributions in the sum or mean graph-pooling by their probabilities. For GTs, we additionally use them to bias the attention scores: $\hat{w}_{ij} = w_{ij} + \log(p_j)$.



(a) GRIT. (b) Graphormer. (c) SAN. (d) GCN.

Figure 3. CLUSTER attack results.

5. Results

Fig. 1 shows the strongest attacks on the different GNNs for CLUSTER and the UPFD fake news detection datasets. It shows that some GTs can be surprisingly fragile. More detailed results are reported in Appendix E.

Results for three attack types are shown: adaptive, transfer, and random. *Adaptive* is the gradient-based PRBCD attack with our relaxations described in § 3. *Random* is a baseline attack that randomly flips as many edges as the budget allows. The random attack gets the same number of model evaluations as the adaptive attack during optimization. Of those, the strongest random perturbation is chosen. For a *transfer* attack, the perturbed graph generated by an adaptive attack on one model is used as an adversarial input for a different GNN model. We report results for the strongest transfer perturbation at each budget.

We evaluate our structure attacks on the inductive node classification dataset CLUSTER [19]. It contains SBM-generated graphs with 6 clusters, where each cluster has one labeled node. The task is to predict which node belongs to which cluster. The node classification accuracy for the CLUSTER dataset for different attack budgets is shown in Fig. 3. We hypothesize that the straightforward nature of the task and dataset lead to the same type of perturbations, independent of the model. This would explain the strong transferability and little variation in robustness across models. This outcome positively indicates the effectiveness of our adaptive attacks, as they consistently identify meaningful perturbations across all GTs. They may sometimes be weaker than the transfer attacks simply due to a more difficult optimization function. To avoid the natural fragility in the data, we also evaluate a constrained attack that prohibits modifying edges to the labeled nodes.

Additionally, we attack the CLUSTER graphs with different combinations of our relaxations. The results for Graphormer and SAN are shown in Tab. 1 and 2 respectively. All continuous relaxations individually seem to provide somewhat useful gradients and can be used to get better than random results. Yet some lead to stronger attacks than others, and using multiple does not seem to work better than just one.

Table 1. Graphormer attacks using relaxations for a fixed budget of 1% for CLUSTER unconstrained (u.) and constrained (c.).

Deg.	SPD	Acc. (%)	
		CLUSTER, u.	CLUSTER, c.
✓	✓	52.61 ± 0.57	60.00 ± 0.42
✓		46.78 ± 0.46	68.45 ± 0.37
	✓	50.81 ± 0.41	60.66 ± 0.21
random		66.52 ± 0.61	70.29 ± 0.32
clean		77.89	77.89

Table 2. SAN attacks using relaxations for a fixed budget of 1% for CLUSTER unconstrained (u.) and constrained (c.).

Atten.	Lap. pert.	Acc. (%)	
		CLUSTER, u.	CLUSTER, c.
✓	✓	53.95 ± 0.59	63.27 ± 0.27
✓		53.87 ± 0.26	63.20 ± 0.12
	✓	57.12 ± 0.62	67.22 ± 0.17
random		65.70 ± 0.65	68.86 ± 0.32
clean		76.12	76.12

Since there is not one that is consistently better than the others, a good approach might be to try all combinations as an ensemble and choose the strongest. A similar evaluation for NIAs on the UPFD datasets is shown in Appendix E. It indicates that only adding the node probability bias to the attention scores seems to be sufficient and leads to some of our strongest attacks.

In conclusion, we empirically demonstrate that GTs can be catastrophically fragile in some settings and remarkably robust w.r.t. the studied attack in other settings. This diverse picture underlines the importance and need for adaptive attacks to reveal nuanced robustness properties. Similarly, also the comparison of GT's and MPNN's robustness w.r.t. the studied attacks does not allow for a conclusion about which approach is superior in terms of robustness. Nevertheless, our work sets the important cornerstone for empirical research in answering this very question.

Acknowledgement

This paper has been supported by the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research.

References

[1] D. Zügner, A. Akbarnejad, and S. Günnemann, “Adversarial attacks on neural networks for graph data,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul. 2018, pp. 2847–2856.

[2] D. Zügner and S. Günnemann, “Adversarial attacks on graph neural networks via meta learning,” in *International Conference on Learning Representations*, 2019.

[3] D. Zügner and S. Günnemann, “Certifiable robustness of graph convolutional networks under structure perturbations,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2020, pp. 1656–1665.

[4] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations*, 2017.

[5] L. Müller, M. Galkin, C. Morris, and L. Rampášek, “Attending to graph transformers,” *Transactions on Machine Learning Research*, Feb. 2024.

[6] A. Athalye, N. Carlini, and D. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” in *Proceedings of the 35th International Conference on Machine Learning*, Jul. 2018, pp. 274–283.

[7] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *Proceedings - IEEE Symposium on Security and Privacy*, 2017, pp. 39–57.

[8] F. Tramèr, N. Carlini, W. Brendel, and A. Madry, “On adaptive attacks to adversarial example defenses,” in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

[9] F. Mujkanovic, S. Geisler, S. Günnemann, and A. Bojchevski, “Are defenses for graph neural networks robust?” In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

[10] L. Ma, C. Lin, D. Lim, *et al.*, “Graph inductive biases in transformers without message passing,” in *Proceedings of the 40th International Conference on Machine Learning*, 2023, pp. 23 321–23 337.

[11] C. Ying, T. Cai, S. Luo, *et al.*, “Do transformers really perform bad for graph representation?” In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

[12] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, and P. Tossou, “Rethinking graph transformers with spectral attention,” in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

[13] K. Xu, H. Chen, S. Liu, *et al.*, “Topology attack and defense for graph neural networks: An optimization perspective,” in *IJCAI’19: Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Jun. 2019, pp. 3961–3967.

[14] S. Geisler, T. Schmidt, H. Şirin, D. Zügner, A. Bojchevski, and S. Günnemann, “Robustness of graph neural networks at scale,” in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021.

[15] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 6000–6010.

[16] G. W. Stewart and J. Sun, *Matrix Perturbation Theory*. 1990, ISBN: 9780126702309.

[17] B. Bamieh, *A tutorial on matrix perturbation theory (using compact matrix notation)*, 2022. arXiv: `2002.05001 [math.SP]`.

[18] J. Wang, M. Luo, F. Suya, J. Li, Z. Yang, and Q. Zheng, “Scalable attack on graph data by injecting vicious nodes,” *Data Min. Knowl. Discov.*, vol. 34, no. 5, pp. 1363–1389, Sep. 2020.

[19] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, “Benchmarking graph neural networks,” *Journal of Machine Learning Research*, vol. 24, no. 43, pp. 1–48, 2023.

[20] H. Dai, H. Li, T. Tian, *et al.*, “Adversarial attack on graph structured data,” in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[21] W. Jin, Y. Li, H. Xu, *et al.*, “Adversarial attacks and defenses on graphs: A review, a tool and empirical studies,” *SIGKDD Explor. Newsl.*, vol. 22, pp. 19–34, 2 2021.

[22] S. Günnemann, “Graph neural networks: Adversarial robustness,” in *Graph Neural Networks: Foundations, Frontiers, and Applications*, L. Wu, P. Cui, J. Pei, and L. Zhao, Eds. 2022, pp. 149–176, ISBN: 9789811660542.

[23] Y. Zhu, J. Huang, Y. Chen, R. Amor, and M. Witbrock, “A graph transformer defence against graph perturbation by a flexible-pass filter,” *Information Fusion*, vol. 107, Jul. 2024.

[24] L. Gosch, D. Sturm, S. Geisler, and S. Günnemann, “Revisiting robustness in graph machine learning,” in *The Eleventh International Conference on Learning Representations*, 2023.

[25] L. Lin, E. Blaser, and H. Wang, “Graph structural attack by perturbing spectral distance,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2022, pp. 989–998.

[26] D. Zhu, P. Cui, Z. Zhang, J. Pei, and W. Zhu, “High-order proximity preserved embedding for dynamic networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, pp. 2134–2144, 11 Nov. 2018.

[27] A. Bojchevski and S. Günnemann, “Adversarial attacks on node embeddings via graph poisoning,” in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 695–704.

[28] H. Wang, Y. Dou, C. Chen, L. Sun, P. S. Yu, and K. Shu, “Attacking fake news detectors via manipulating news social engagement,” in *ACM Web Conference 2023 - Proceedings of the World Wide Web Conference, WWW 2023*, 2023, pp. 3978–3986.

[29] X. Zou, Q. Zheng, Y. Dong, *et al.*, “TDGIA: Effective injection attacks on graph neural networks,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2021, pp. 2461–2471.

[30] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, “Recipe for a general, powerful, scalable graph transformer,” in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, 2022.

[31] J. You, R. Ying, and J. Leskovec, “Design space for graph neural networks,” in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

[32] M. Fey and J. E. Lenssen, *Fast graph representation learning with PyTorch Geometric*, 2019. arXiv: 1903.02428 [cs.LG].

[33] Y. Dou, K. Shu, C. Xia, P. S. Yu, and L. Sun, “User preference-aware fake news detection,” in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2051–2055.

[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” in *International Conference on Learning Representations*, 2018.

[35] S. Brody, U. Alon, and E. Yahav, “How attentive are graph attention networks?” In *International Conference on Learning Representations*, 2022.

A. Related work

Triggered by the seminal works of Zügner, Akbarnejad, and Günnemann [1] and Dai, Li, Tian, *et al.* [20], a research area emerged spanning attacks, defenses, and certification of GNNs [21], [22]. However, GTs have been entirely neglected despite their recent success on common benchmarks. Zhu, Huang, Chen, *et al.* [23] is the sole exception acknowledging this research gap. However, they solely propose a robust and sparse transformer and evaluate it with non-adaptive poisoning attacks. Thus, they do not shine light on the robustness of the diverse set of graph transformers nor do they study adaptive attacks.

Our attack is rooted in the GNN robustness literature. Xu, Chen, Liu, *et al.* [13] proposes the first Projected Gradient Descent attack for discrete L0 perturbations of the graph structure, with a focus on message-passing architectures. Geisler, Schmidt, Şirin, *et al.* [14] extend this PGD with a randomization scheme to obtain the efficient Projected Randomized Block Coordinate (PRBCD) attack. Gosch, Sturm, Geisler, *et al.* [24] extend PRBCD with local constraints, which is comparable with our relaxed GTs. While highlighting the general nature of our framework, we leave the empirical evaluation for future work. Further important related works are Lin, Blaser, and Wang [25], Zhu, Cui, Zhang, *et al.* [26], and Bojchevski and Günnemann [27], where the authors study similar approximations for perturbations on the eigen-decomposition of the graph Laplacian. Moreover, Wang, Dou, Chen, *et al.* [28] attack message-passing architectures on the UPFD fake news detection using reinforcement learning. As an entry to Node Insertion Attacks (NIA), we refer to Wang, Luo, Suya, *et al.* [18] and Zou, Zheng, Dong, *et al.* [29]

B. Graph transformers

Graph transformers (GTs) apply the popular transformer architecture for sequences [15] to arbitrary graphs. A general GT architecture is depicted in Fig. 2. In this work, we focus on GTs that apply global self-attention, where each node can attend to all other nodes. A 'vanilla' structure-unaware self-attention head is defined as:

$$\mathsf{Attn}(\boldsymbol{X}) = \mathsf{softmax}\left(\frac{(\boldsymbol{X}\boldsymbol{W}_q)(\boldsymbol{X}\boldsymbol{W}_k)^\mathsf{T}}{\sqrt{d_k}}\right)(\boldsymbol{X}\boldsymbol{W}_v) \tag{11}$$

where $\boldsymbol{W}_q, \boldsymbol{W}_k \in \mathbb{R}^{d\times d_k}$ are the weights for the *query* and *key* projections, and $\boldsymbol{W}_v \in \mathbb{R}^{d\times d}$ is the *value* weight matrix. For structure awareness, prior works commonly add Positional Encodings (PEs) to the node features:

$$\hat{\boldsymbol{X}} = \boldsymbol{X} + \psi(\boldsymbol{A}) \tag{12}$$

GT categorization. We categorize the PEs roughly in three main categories: (1) random walk encodings, (2) distance encodings, and (3) spectral encodings. In addition to PEs, many GTs adapt the attention mechanism to capture relevant topological aspects of the graph explicitly in the attention operation. Some GT architectures can optionally leverage available edge features, which we omit here since our evaluation data does not include any. Furthermore, edge features are typically not considered for pure structure perturbation attacks, as adding new edges would also require adding features for them. Next, we introduce a representative GT for each of the PE categories.

(1) Random-walk-based GRIT. The GRIT model architecture [10] terms their PE Relative Random Walk Probabilities (RRWP). The random walk encodings are collected from a fixed-length walk of length k, which is a hyperparameter (usually $k > 10$). The PEs are based on the tensor:

$$\boldsymbol{P} = [\boldsymbol{I}, \boldsymbol{M}, \boldsymbol{M}^2, ..., \boldsymbol{M}^{k-1}] \in \mathbb{R}^{n\times n\times k}, \quad \text{with} \quad \boldsymbol{M} = \boldsymbol{D}^{-1}\boldsymbol{A} \tag{13}$$

This yields an embedding vector $\boldsymbol{P}_{ij} \in \mathbb{R}^k$ for each of the n^2 node-pairs (v_i, v_j). The diagonal vector entries are transformed to dimension d by a linear layer and added to the node features as PEs: $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i + g_1(\boldsymbol{P}_{ii})$. Additionally, all n^2 vectors are transformed by a separate linear layer and added as node-pair features: $\boldsymbol{h}_{i,j}^{(0)} = g_2(\boldsymbol{P}_{ij})$. The node representations $\boldsymbol{h}_i$ and node-pair representations $\boldsymbol{h}_{i,j}$ are updated in each transformer layer by a modified attention mechanism, which includes an adaptive degree-scaler that is applied to the node representations:

$$\boldsymbol{h}_i = (\boldsymbol{h}_i \odot \boldsymbol{\theta}_1) + \log(1 + \mathsf{deg}(v_i)) \cdot (\boldsymbol{h}_i \odot \boldsymbol{\theta}_2) \tag{14}$$

where $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$ are learnable weights.

(2) Distance-based Graphormer. The Graphormer model [11] uses degree PEs. For each discrete degree value there is a learnable embedding vector $\boldsymbol{z} \in \mathbb{R}^d$. The embeddings are added to the node features according to the node degrees:

$$\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i + \boldsymbol{z}_{\mathsf{deg}(v_i)} \tag{15}$$

Similarly, a learnable scalar $b \in \mathbb{R}$ is assigned to each discrete Shortest Path Distance (SPD). This value is added to the raw attention scores and results in a re-weighting of the attention weights after applying the softmax function:

$$\hat{w_{ij}} = w_{ij} + b_{\mathsf{spd}(v_i, v_j)}, \qquad \alpha_{ij} = \mathsf{softmax}(\hat{w}_{ij}) \tag{16}$$

where $w_{ij} = \boldsymbol{W}_q\boldsymbol{h}_i \cdot \boldsymbol{W}_k\boldsymbol{h}_j / \sqrt{d}$. For graph-level tasks, a virtual node is added to the graph with its own distinct learnable bias b_{virtual} , which is used as graph representation in the pooling stage.

(3) Spectral SAN The SAN architecture [12] uses learned Laplacian-based PEs (LPEs), starting with the eigen-decomposition of the Laplacian $\boldsymbol{L}_{sym} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, where diagonal entries of $\boldsymbol{\Lambda}_{ii} = \lambda_i$ are the eigenvalues of $\boldsymbol{L}_{sym}$ in ascending order $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$, and the columns of $\boldsymbol{U}$ are the corresponding eigenvectors. Determined by a hyperparameter, only the k-th smallest eigenvalues and their eigenvectors are used, which we write as $\boldsymbol{\Lambda}_k \in \mathbb{R}^{k \times k}$ and $\boldsymbol{U}_k \in \mathbb{R}^{n \times k}$. For each node v_i, its PEs are initialized as the concatenation of the eigenvalues and the i-th row of $\boldsymbol{U}_k$:

$$\boldsymbol{P}_i = [\mathsf{diag}(\boldsymbol{\Lambda}_k) \parallel (\boldsymbol{U}_k)_i] \in \mathbb{R}^{k \times 2} \tag{17}$$

Further processing by a transformer encoder results in $\boldsymbol{p}_i = f(\boldsymbol{P}_i) \in \mathbb{R}^{d_p}$, which is concatenated to the node features: $\boldsymbol{h}_i^{(0)} = \boldsymbol{x}_i \parallel \boldsymbol{p}_i$. Additionally, the main graph transformer attention mechanism is modified to have two separate key and query weights for connected and unconnected node-pairs. The attention scores to the connected nodes and to the unconnected nodes are computed independently, each with a softmax. A hyperparameter $\gamma \in \mathbb{R}^+$ controls how the two scores are relatively scaled, varying the bias towards sparse or full attention:

$$\alpha_{ij} = \begin{cases} \frac{1}{1+\gamma}\mathsf{softmax}\left(\boldsymbol{W}_{q,\mathsf{real}}\boldsymbol{h}_i \cdot \boldsymbol{W}_{k,\mathsf{real}}\boldsymbol{h}_j / \sqrt{d}\right) & \text{if } (v_i, v_j) \text{ is a real edge} \\ \frac{\gamma}{1+\gamma}\mathsf{softmax}\left(\boldsymbol{W}_{q,\mathsf{fake}}\boldsymbol{h}_i \cdot \boldsymbol{W}_{k,\mathsf{fake}}\boldsymbol{h}_j / \sqrt{d}\right) & \text{otherwise} \end{cases} \tag{18}$$

Some graph transformer architectures (including all three chosen in this work) can optionally leverage available edge features, which we omit here since our evaluation data does not include any. Furthermore, edge features are typically not considered for pure structure perturbation attacks, as adding new edges would also require adding features for them.

C. Node injection attack

We also consider the relevant case of inserting nodes into an existing graph structure. In contrast to the usual framing of Node Injection Attack (NIA), where the attacker also chooses the node features for the new *vicious* nodes [18], we connect existing nodes from other graphs of an inductive graph dataset. Therefore, the nodes' features are fixed but physically realizable even if, e.g., they represent embeddings of natural language. NIA in this inductive setting is also interesting for attacking GTs since they allow to increase the graph size successively. This growth implies that the attention mechanism must allow for a smooth insertion of nodes.

Let $\mathcal{D} = \{\mathcal{G}_1, ..., \mathcal{G}_N\}$ be the dataset including all graphs, where each graph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ has n_i nodes $V_i = \{v_{i,1}, ..., v_{i,n_i}\}$ with node feature matrix $\boldsymbol{X}_i \in \mathbb{R}^{n_i \times d}$. The total number of nodes in the dataset is $n_\mathcal{D} = \sum n_i$. Let $\mathcal{G}_{atk}$ be the graph that is being attacked. We define the candidate set of injection nodes as the union of the nodes of all other graphs: $\mathcal{V}_{cs} = \bigcup_{\mathcal{G}_i \in \mathcal{D} \setminus \mathcal{G}_{atk}} \mathcal{V}_i$, which includes $n_{cs} = n_\mathcal{D} - n_{atk}$ nodes with corresponding features $\boldsymbol{X}_{cs}$. It is of course possible to restrict this candidate set if is is not sensible or not feasible to include all nodes.

We can augment the original (connected) graph by adding the injection candidate set as isolated nodes. Then $\mathcal{G}'_{atk} = (\boldsymbol{A}'_{atk}, \boldsymbol{X}'_{atk})$ is the augmented graph with:

$$\boldsymbol{A}'_{atk} = \begin{bmatrix} \boldsymbol{A}_{atk} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \in \{0,1\}^{n_\mathcal{D} \times n_\mathcal{D}}, \qquad \boldsymbol{X}'_{atk} = \begin{bmatrix} \boldsymbol{X}_{atk} \\ \boldsymbol{X}_{cs} \end{bmatrix} \in \mathbb{R}^{n_\mathcal{D} \times d} \tag{19}$$

Edge-flip perturbations to this augmented adjacency matrix, $\tilde{\boldsymbol{A}}' = \boldsymbol{A}' + \delta\boldsymbol{A}'$, model both structure perturbations and node injections together. As in Eq. 2, the perturbation $\delta\boldsymbol{A}'$ can be expressed in terms of a binary edge flip matrix:

$$\tilde{\boldsymbol{A}}' = \boldsymbol{A}' + (\boldsymbol{1}_n\boldsymbol{1}_n^\top - 2\boldsymbol{A}') \odot \boldsymbol{B}', \qquad \text{with} \quad \boldsymbol{B}' = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{E} \\ \boldsymbol{E}^\top & \boldsymbol{F} \end{bmatrix} \in \{0,1\}^{n_\mathcal{D} \times n_\mathcal{D}} \tag{20}$$

Since the number of edge flips is bounded by a budget usually much smaller than the candidate set size i.e., $\Delta \ll n_{cs}$, the perturbed augmented graph $\tilde{\mathcal{G}} = (\tilde{\boldsymbol{A}}', \boldsymbol{X}')$ still mostly contains isolated nodes. Therefore, we prune away all disconnected components, which for the unperturbed graph simply reverts the augmentation: $\mathsf{prune}(\mathcal{G}') = \mathcal{G}$. However, for a perturbed augmented graph, this results in the perturbed graph that we are seeking:

$$\tilde{\mathcal{G}} = (\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{X}}) = \mathsf{prune}(\tilde{\boldsymbol{A}}', \boldsymbol{X}') \tag{21}$$

where, $\tilde{\boldsymbol{A}} \in \{0,1\}^{\tilde{n}\times\tilde{n}}$, $\tilde{\boldsymbol{X}} \in \mathbb{R}^{\tilde{n}\times d}$, $\tilde{n} = n + n_{in}$ is the total number of nodes and n_{in} is the number of new injected nodes.

The node injection attack objective can thus be written as:

$$\max_{\boldsymbol{B}' \text{ s.t. } \|\boldsymbol{B}'\|_0 < \Delta} \mathcal{L}_{atk}(f_\theta(\tilde{\mathcal{G}})) \tag{22}$$

where, f_θ is the trained GNN and $\mathcal{L}_{atk}$ is a suitable attack loss. Note that the edge flip budget Δ is also an upper bound for the number nodes that can be injected: $0 \le n_{in} \le \Delta$.

Edge block sampling. To optimize the objective, we can apply the relaxation $\boldsymbol{B}_{ij} \in [0,1]$, as shown in § 2. In this case, PRBCD [14] not only enables more efficient optimization, but setting a smaller block size is crucial to limit the number of connected injection nodes during optimization, since GTs complexity scales with $O(\tilde{n}^2)$. Moreover, the edge sampling allows us to control which parts of $\boldsymbol{B}'$ in Eq. 20 to sample from, e.g. not sampling in $\boldsymbol{B}$ results in 'pure' node injections without modifying edges in the original graph. For large candidate sets, we recommend only sampling from $\boldsymbol{E}$, as sampling from the n_{cs}^2 entries of $\boldsymbol{F}$ results in many disconnected injection node pairs that get pruned away.

Node probability attention bias. The continuous optimization of structure attacks in § 2 assigns probabilities to edges-flips, while nodes are assumed to all be part of the graph. In contrast, during NIAs the nodes also have have certain probabilities of being included. To obtain node probabilities from the edge weights in a general way, we propose a simple iterative computation. For a continuous connected graph, let $\mathcal{N}(v_i)$ be the 1-hop neighborhood of node v_i. We can approximate the probability p_i of v_i being connected to the graph, by using the probability that it is connected to its neighbors and the probability that these neighbors are connected to the graph. We start with the assumption that all nodes are connected to the graph and update using the edge probabilities:

$$p_i^{(t+1)} = 1 - \prod_{v_j \in \mathcal{N}(v_i)} (1 - \boldsymbol{A}_{ij} \cdot p_j^{(t)}), \quad \text{with} \quad p_i^{(0)} = 1 \tag{23}$$

The number of iterations should be set to match the expected longest chain of added injection nodes. Therefore very few iterations (2-3) should suffice for most NIAs.

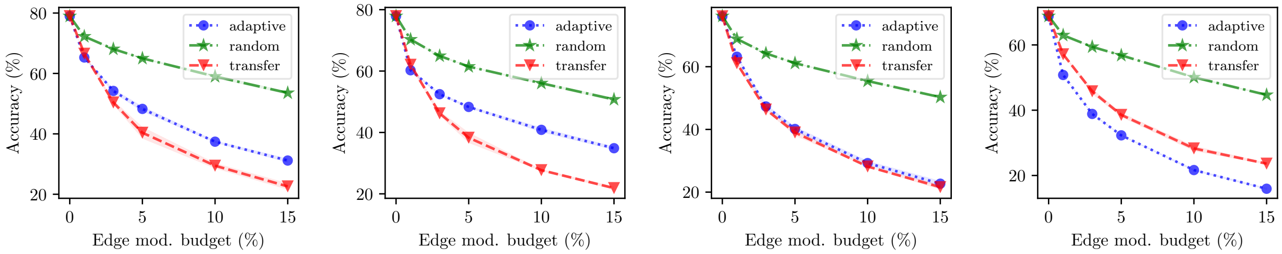
We use the node probability to compute a weighted sum or mean in the graph-pooling for graph level tasks, to ensure that the model output is continuous w.r.t. node injections. Additionally for GTs, we use the node probability to bias the global pairwise attention scores which result a continuous weighting of the attention scores:

$$\hat{w}_{ij} = w_{ij} + \log(p_j), \qquad \hat{\alpha}_{ij} = \mathsf{softmax}(\hat{w}_{ij}) = \frac{e^{\hat{w}_{ij}}}{\sum_k e^{\hat{w}_{ik}}} = \frac{p_j \cdot e^{w_{ij}}}{\sum_k p_k \cdot e^{w_{ik}}} \tag{24}$$

D. Evaluation details

Setup. We build our implementation on top of GPS [30], which uses the flexible GraphGym [31] framework of the PyG [32] library. All our experiments ran on an internal GPU cluster. Experiments that require less than 10GB of GPU memory were conducted on a Nvidia GTX1080Ti (10GB), the others on V100 and A100 GPUs (32/40GB). Attacking a single graph in our experiments takes anywhere from a few seconds to a few minutes, depending on the graph and model sizes.

Datasets. We evaluate our structure attacks on the inductive node classification dataset CLUSTER [19]. It contains SBM-generated graphs with 6 clusters, where each cluster has one labeled node. The average number of nodes is 117. The task is to predict which node belongs to which cluster. For training, we use the standard PyG train/val/test split of 10000/1000/1000 graphs, respectively. We use the UPFD Twitter fake news detection datasets from [33] to evaluate our node injection attacks. There are 2 datasets: *politifact*, with political; and *gossipcop* with celebrity fake news. The average number of nodes is 131 and 58, respectively. The graphs consist of retweet trees, where the root node has features concerning both the news content and the user who posted the news. All other nodes' features are related to the users that retweeted the news. The task is



(a) GRIT.

(b) Graphormer.

(c) SAN.

(d) GCN.

Figure 4. CLUSTER constrained attack results.

binary classification whether the graph contains fake news or not. We use the standard PyG train/val/test split of 20/10/70% of the $314/5464$ graphs for politifact and gossipcop, respectively.

Due to the quadratic scaling in the number of nodes of the three chosen GTs, their application is limited to smaller graphs. This renders evaluation on larger graph datasets commonly used in robustness studies impractical. While GTs are most widely applied to molecule data, adversarial attacks are of little practical relevance in that domain. Thus, we omit molecule data from our evaluations.

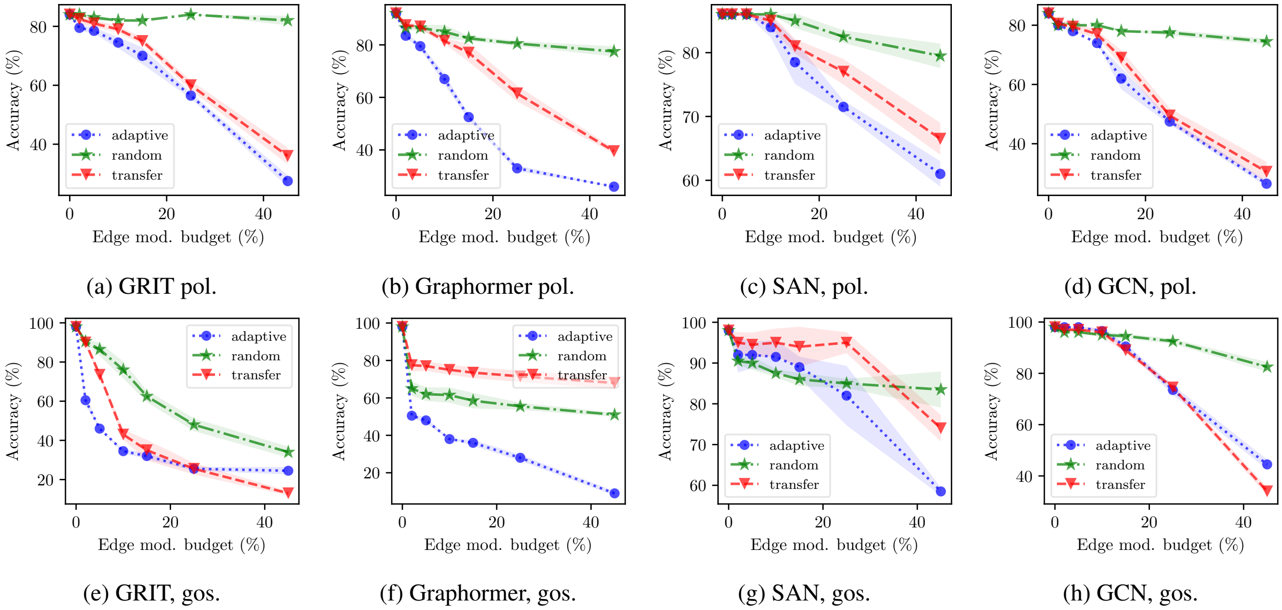
Attacks. In this work we focus on *untargeted* white-box *evasion* attacks, i.e., an attacker with full knowledge of model and data attempts to change the trained model's prediction to any incorrect class at test time by slightly perturbing the input graph structure. For node-level tasks we focus on *global* attacks that minimize the overall performance metric across all nodes. For model training we do a random hyperparameter search, choosing the model with the highest validation metric. This approach is consistent with common practice. As described in § 5, we show results for three attack types: *adaptive*, *transfer*, and *random* for a range of different budgets.

For both CLUSTER and the UPFD datasets, we evaluate our attacks on the 50 first graphs in the test set and report average and standard deviation over 4 random seeds. For node classification, we use *tanh-margin* attack loss, shown to be effective in [14]. For the binary graph classification, we simply minimize/maximize the raw prediction score (before sigmoid) for labels 1/0 as our attack loss. For CLUSTER attacks, we use a PRBCD block size of 20000. For UPFD, we use a small block size of 1000, which is necessary due to the n^2 scaling of GTs. For GCN it is possible to increase the block size, but we keep it the same for comparability. We optimize all our adaptive attacks for 125 steps and sample 20 discrete perturbations from the result, of which we take the strongest. For all other attack hyperparameters, we use default values that performed well in preliminary evaluations. For all results except the ablations, we use all of our continuous relaxations proposed in § 3.

E. Additional results

We present the first principled analysis on the robustness of GTs on three representative architecture types (GRIT, Graphormer, SAN). We define different goals for our evaluation: **(A)** efficacy of the proposed adaptive attacks, **(B)** providing an accurate assessment of GT robustness for relevant real-world tasks. To this end, we perform our evaluation on datasets with varying complexity. Towards **(A)** we explore the robustness of GTs on CLUSTER, which comprises simple, interpretable structures. This exploration helps us evaluate the effectiveness of the proposed relaxations, ideally leading our attacks to target semantically meaningful structures within the dataset. We address **(B)** through evaluations on UPFD. Here, we constrain our attack to remain within the predefined tree structure of the dataset. As a result, the attack represents impersonating an existing user who is retweeting the respective news article. This evaluation goes beyond previous robustness analyses of citation networks in GNNs [1], [14], offering a more practical use case and semantically meaningful attacks.

CLUSTER. The node classification accuracy for the CLUSTER dataset for different attack budgets is shown in Fig. 3. Since only a single node in each cluster is labeled, attacking these labeled nodes requires little budget and leads to strong attacks. We manually inspected the perturbations and confirmed that most edge modifications are connected to the labeled nodes, which shows the efficacy of the attacks. The difference in attack strength between the adaptive and transfer attacks is insignificant. This confirms our hypothesis that the straightforward nature of the task leads to the same type of semantically meaningful perturbations, independent of the model (modifying the edges to labeled nodes). This would explain the strong transferability and little variation in robustness across models. This outcome positively indicates the effectiveness of our adaptive attacks **(A)**, as they consistently identify meaningful perturbations across all GTs. They may sometimes be weaker



(a) GRIT pol.

(b) Graphormer pol.

(c) SAN, pol.

(d) GCN, pol.

(e) GRIT, gos.

(f) Graphormer, gos.

(g) SAN, gos.

(h) GCN, gos.

Figure 5. Attack results for the UPFD twitter datasets politifact (pol.) and gossipcop (gos.).

than the transfer attacks simply due to a more difficult optimization function (GCN has fewer non-differentiable components). To avoid the natural fragility in the data, we also evaluate a constrained attack that prohibits modifying edges to the labeled nodes, for which results are shown in Fig. 4. For CLUSTER, we additionally attack the GAT [34] and GATv2 [35] GNN models and include the adversarial perturbations for these in the transfer attack. Results for these are also shown in Fig 1.

UPFD. For our node injection attack, we add all dataset nodes except for the graph roots into the candidate set since they have unique properties and include user and news features. We do not allow for perturbations of the original tree structure. If the discretely sampled perturbations do not have a tree structure, we take the maximum spanning tree (using the edge probabilities) to ensure all perturbations are valid retweet trees.

The graph classification accuracy for different attack budgets is shown in Fig. 5. In contrast to the results observed for the CLUSTER dataset, we note big robustness differences across models for UPFD. In general, Graphormer seems to be the least robust and for which our adaptive attacks work best in comparison the baselines. For Graphormer on gossipcop the random baseline creates significantly stronger attacks than the transfer attacks, highlighting the importance of adaptive attacks. The adaptive attacks for GRIT are also significantly stronger than the baselines. The SAN attacks are the outlier, and for gossipcop the random baseline generates significantly stronger attacks than both adaptive and transfer. The reason may be that the Laplacian eigen-decomposition perturbation approximation does not hold well in the case of node injection. The ablation in Tab. 4 also indicates that by disabling this relaxation much stronger attacks than reported in Fig. 5 are possible.

Our results reveal that GTs showcase catastrophic vulnerabilities to adversarial modifications of the graph structure, even when these changes are constrained to meaningful perturbations. In comparison, the GCN model exhibits considerably higher robustness. Furthermore, we cannot derive a consistent pattern concerning the robustness of the different GT architectures and results differ depending on the dataset and attack. Both findings underscore the need for research into robust design choices of architecture components, such as PEs, and modifications to the attention mechanism.

Ablations. We enable each of the continuous relaxations individually and in different combinations together. We report the results for Graphormer and SAN in Tab. 3 and 4 respectively. The node probability relaxation only applies to the node injection attacks on UPFD. The main insights from the results are: **(a)** all continuous relaxations individually seem to give somewhat useful gradients and can be used to get better than random results. **(b)** some work better than others, and using multiple does not seem to always work better than only one. We argue that they lead the optimization towards different adversarial perturbations, targeting features relevant for different components. However, one is not consistently better than the other. **(c)** adding the node probability bias to the attention scores for node injection results is usually sufficient and leads to some of the strongest attacks. **(d)** A good approach might be to try all individually as an ensemble and choose the strongest. Additionally, we check the attack strength for GRIT when enabling or disabling gradient computation through

Table 3. Ablations for the Graphormer relaxations for a fixed budget of 1% for CLUSTER unconstrained (u.) and constrained (c.), and 10% for UPFD politifact (pol.) and gossipcop (gos.). Mean and standard deviation over 4 runs with different seeds.

Deg.	SPD	Acc. (%)		Node prob.	Acc. (%)	
		CLUSTER, u.	CLUSTER, c.		UPFD pol.	UPFD gos.
✓	✓	52.61 ± 0.57	**60.00** ± 0.42	✓	67.0 ± 2.0	**38.0** ± 0.0
✓		**46.78** ± 0.46	68.45 ± 0.37	✓	67.0 ± 2.0	**38.0** ± 0.0
	✓	50.81 ± 0.41	60.66 ± 0.21	✓	**66.5** ± 1.9	39.5 ± 1.9
				✓	**66.5** ± 1.9	38.5 ± 1.0
✓	✓				80.5 ± 3.4	53.5 ± 1.0
random		66.52 ± 0.61	70.29 ± 0.32		85.0 ± 2.6	61.5 ± 4.1
clean		77.89	77.89		92.0	98.0

Table 4. Ablations for the SAN relaxations for a fixed budget of 1% for CLUSTER unconstrained (u.) and constrained (c.), and 10% for UPFD politifact (pol.) and gossipcop (gos.). Mean and standard deviation over 4 runs with different seeds.

Atten.	Lap. pert.	Acc. (%)		Node prob.	Acc. (%)	
		CLUSTER, u.	CLUSTER, c.		UPFD pol.	UPFD gos.
✓	✓	53.95 ± 0.59	63.27 ± 0.27	✓	83.5 ± 1.0	91.5 ± 4.1
✓		**53.87** ± 0.26	**63.20** ± 0.12	✓	77.5 ± 1.0	91.5 ± 2.5
	✓	57.12 ± 0.62	67.22 ± 0.17	✓	83.5 ± 1.0	89.5 ± 1.9
				✓	**77.0** ± 1.2	89.5 ± 3.4
✓	✓				86.0 ± 0.0	90.1 ± 6.0
random		65.70 ± 0.65	68.86 ± 0.32		86.0 ± 0.0	**87.5** ± 1.0
clean		76.12	76.12		86.0	98.0

certain parts of the model and show the results in Tab. 5. It is possible to get strong attacks even without computing gradients through RRWP, which could be much more efficient computationally, depending on the model and graph size. For node injection attacks, as for the other models, using only the node probability bias in the attention scores already leads to the strongest attacks we report.

F. Limitations

All our adversarial attacks do neither provide a guarantee nor insights about how close they are to the optimal adversarial example. We note that this is common practice in the adversarial learning community. Nevertheless, if we are able to find an adversarial example, this proves the non-robustness of the studied model. Furthermore, we only look at three specific graph transformer architectures. While the continuous relaxations described here may not always be directly applicable to other GTs, our approach for designing adaptive attacks should provide a useful guide for further relaxations of similar components.

Table 5. Ablations for the GRIT relaxations for a fixed budget of 1% for CLUSTER unconstrained (u.) and constrained (c.), and 10% for UPFD politifact (pol.) and gossipcop (gos.). Mean and standard deviation over 4 runs with different seeds.

RRWP grad.	Deg. grad.	Acc. (%)		Node prob.	Acc. (%)	
		CLUSTER, u.	CLUSTER, c.		UPFD pol.	UPFD gos.
✓	✓	**44.07** ± 0.79	**65.25** ± 0.22	✓	**34.5** ± 1.0	75.0 ± 2.6
✓		46.27 ± 0.36	65.70 ± 0.35	✓	**34.5** ± 1.0	74.5 ± 2.5
	✓	49.51 ± 0.90	66.49 ± 0.49	✓	**34.5** ± 1.0	**73.5** ± 1.0
				✓	**34.5** ± 1.0	**73.5** ± 1.0
✓	✓				54.5 ± 1.9	83.0 ± 2.0
random		69.13 ± 0.10	72.25 ± 0.29		76.0 ± 4.3	82.0 ± 0.0
clean		78.98	78.98		98.0	84.0