Adversarial Attacks on Graph Classification via Bayesian Optimisation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Graph neural networks, a popular class of models effective in a wide range of graph-based learning tasks, have been shown to be vulnerable to adversarial attacks. While the majority of the literature focuses on such vulnerability in node-level classification tasks, little effort has been dedicated to analysing adversarial attacks on graph-level classification, an important problem with numerous real-life applications such as biochemistry and social network analysis. The few existing methods often require unrealistic setups, such as access to internal information of the victim models, or an impractically-large number of queries. We present a novel Bayesian optimisation-based attack method for graph classification models. Our method is *black-box*, *query-efficient* and *parsimonious* with respect to the perturbation applied. We empirically validate the effectiveness and flexibility of the proposed method on a wide range of graph classification tasks involving varying graph properties, constraints and modes of attack. Finally, we analyse common interpretable patterns behind the adversarial samples produced, which may shed further light on the adversarial robustness of graph classification models.

1 Introduction

Graphs are a general-purpose data structure consisting of entities represented by nodes and edges which encode pairwise relationships. In recent years, graph-based machine learning models has been widely used in a variety of important applications such as semi-supervised learning, link prediction, community detection and graph classification [2, 36, 9]. Despite the growing interest in graph-based machine learning, it has been shown that, like many other machine learning models, graph-based models are vulnerable to adversarial attacks [26, 11]. If we want to deploy such models in environments where the risk and costs associated with a model failure are high e.g. in social networks, it would be crucial to understand and assess the model stability and vulnerability by simulating adversarial attacks.

Adversarial attacks on graphs can be aimed at different learning tasks. This paper focuses on graph-level classification, where given an input graph (potentially with node and edge attributes), we wish to learn a function that predicts a property of interest related to the graph. Graph classification is an important task with many real-life applications, especially in bioinformatics and chemistry [18, 19]. For example, the task may be to accurately classify if a molecule, modelled as a graph whereby nodes represent atoms and edges model bonds, inhibits HIV replication or not. Although there are a few attempts on performing adversarial attacks on graph classification [7, 17], they all operate under unrealistic assumptions such as the need to query the target model a large number of times or access a portion of the test set to train the attacking agent.

To address these limitations, we formulate the adversarial attack on graph classification as a black-box optimisation problem and solve it with Bayesian optimisation (BO), a query-efficient state-of-the-

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Algorithm 1 Overall pseudocode of the GRABNEL routine.

Input: Original graph $\mathcal{G}_0$, victim model f_θ, n_{init} (the number of random initialising points), Query budget B, Perturbation budget Δ.
Output: An adversarial graph $\mathcal{G}^*$
Set base graph $\mathcal{G}_{\text{base}} \leftarrow \mathcal{G}_0$; initialise stage count $\texttt{stage} \leftarrow 0$.
Randomly sample n_{init} perturbed graphs $\{\mathcal{G}'\}_{i=1}^{n_{\text{init}}}$ that are 1 edit distance different from $\mathcal{G}$ and query each perturbed graph to obtain their attack losses $\mathcal{L}_{\text{attack}}(f_\theta, \mathcal{G}'_i)$.
Compute the WL feature encoding for all graphs: $(\Phi(\mathcal{G}'_1), \ldots, \Phi(\mathcal{G}'_{n_{\text{init}}})) = \texttt{WLFeatureExtract}(\mathcal{G}_0, (\mathcal{G}'_1, \ldots \mathcal{G}'_{n_{\text{init}}}))$. *// See App. A for details of* `WLFeatureExtract`.
Fit the sparse Bayesian linear regression surrogate with the data $\{\Phi(\mathcal{G}'_i), \mathcal{L}_{\text{attack}}(f_\theta, \mathcal{G}'_i)\}_{i=1}^{n_{\text{init}}}$
Divide total budget of B into Δ stages *// See "Sequential perturbation selection"*
while query budget is not exhausted and attack has not succeeded **do**
  if query budget of the *current stage* is exhausted **then**
    Increment the stage count $\texttt{stage} \leftarrow \texttt{stage} + 1$ and update the base graph $\mathcal{G}_{\text{base}}$ with the graph leading to largest increase in attack loss in the previous stage. *// Refer to Fig. 1*
  end if
  Propose graph to be queried next $\mathcal{G}'_{\text{proposal}}$ via acquisition optimisation. *// See "Optimisation of acquisition function"*
  Query f_θ for the graph proposed in the previous step to calculate its attack loss.
  if attack succeeded **then**
    Set $\mathcal{G}^* \leftarrow \mathcal{G}'_{\text{proposal}}$ and **return** it.
  end if
  Augment the observed data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathcal{G}'_{\text{proposal}}, \mathcal{L}_{\text{attack}}(f_\theta, \mathcal{G}'_{\text{proposal}})\}$, update the WL feature encodings of *all* observed graphs $(\Phi(\mathcal{G}'_1), \ldots, \Phi(\mathcal{G}'_{|\mathcal{D}|})) = \texttt{WLFeatureExtract}(\mathcal{G}_0, (\mathcal{G}'_1, \ldots \mathcal{G}'_{|\mathcal{D}|}))$ and re-fit the surrogate.
end while
return `None` *// Failed attack within the query budget*

art black-box optimiser. Unlike existing work, our method is query efficient, parsimonious in perturbations and does not require supervised training on a labelled dataset to effectively attack a new sample. Another benefit of our method is that it can be easily adapted to perform various modes of attacks such as deleting or rewiring edges and node injection. Furthermore, we investigate the topological properties of the successful adversarial examples found by our method and offer valuable insights on the connection between the graph topology change and the model robustness.

The main contributions of our paper are as follows. First, we introduce a novel black-box attack for graph classification, GRABNEL[1], which is both query efficient and parsimonious. We believe this is the first work on using BO for adversarial attacks on graph data. Second, we analyse the generated adversarial examples to link the vulnerability of graph-based machine learning models to the topological properties of the perturbed graph, an important step towards interpretable adversarial examples that has been overlooked by the majority of the literature. Finally, we evaluate our method on a range of real-world datasets and scenarios including detecting the spread of fake news on Twitter, which to the best of our knowledge is the first analysis of this kind in the literature.

2 Proposed Method: GRABNEL

Problem Setup A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by a set of nodes $\mathcal{V} = \{v_i\}_{i=1}^n$ and edges $\mathcal{E} = \{\mathbf{e}_i\}_{i=1}^m$ where each edge $\mathbf{e}_k = \{v_i, v_j\}$ connects between nodes v_i and v_j. The overall topology can be represented by the adjacency matrix $\mathbf{A} \in \{0,1\}^{n \times n}$ where $\mathbf{A}_{ij} = 1$[2] if the edge $\{v_i, v_j\}$ is present. The attack objective in our case is to degrade the predictive performance of the pretrained victim graph classifier f_θ by finding a graph $\mathcal{G}'$ perturbed from the original test graph $\mathcal{G}$ (ideally with the minimum amount of perturbation) such that f_θ produces an incorrect class label for $\mathcal{G}$. In this paper, we consider the *black-box evasion* attack setting, where the adversary agent cannot access or modify the the victim model f_θ (i.e. its network architecture, its weights θ or gradients) or its training data $\{(\mathcal{G}_i, y_i)\}_{i=1}^L$; the adversary can only interact with f_θ by querying it with an input graph $\mathcal{G}'$ and observe the model output as pseudo-probabilities over all classes $f_\theta(\mathcal{G}') \in [0,1]^C$. Additionally, we assume that *sample efficiency* is highly valued; the number of queries should be as few as possible to avoid detection in a real-life scenario. Formally, the objective can be formulated as a black-box maximisation problem on graph $\mathcal{G}$, which is used as the objective function of our BO attack agent:

$$\max_{\mathcal{G}' \in \Psi(\mathcal{G})} \mathcal{L}_{\text{attack}}\big(f_\theta(\mathcal{G}'), y\big) \;\text{ s.t. }\; y = \arg\max f_\theta(\mathcal{G}) \tag{1}$$

[1] Stands for *Graph Adversarial attack via Bayesian efficient loss-minimisation*.

[2] We discuss the unweighted graphs for simplicity; our method may also handle other graph types.

where f_θ is the pretrained victim model that remains fixed in the evasion attack setup and y is the correct label of the original input $\mathcal{G}$. Denote the output logit for the class y as $f_\theta(\mathcal{G})_y$, the *attack* loss $\mathcal{L}_{\text{attack}}$ can be defined as:

$$\mathcal{L}_{\text{attack}}\Big(f_\theta(\mathcal{G}'), y\Big) = \begin{cases} \max_{t\in\mathcal{Y}, t\neq y} \log f_\theta(\mathcal{G}')_t - \log f_\theta(\mathcal{G}')_y & \text{(untargeted attack)} \\ \log f_\theta(\mathcal{G}')_t - \log f_\theta(\mathcal{G}')_y & \text{(targeted attack on class } t\text{)}, \end{cases} \tag{2}$$

where $f_\theta(\cdot)_t$ denotes the logit output for class t. Such attack loss definition is commonly used both in the traditional image attack and the graph attack literature [3, 37]. Furthermore, $\Psi(\mathcal{G})$ refers to the set of possible $\mathcal{G}'$ generated from perturbing $\mathcal{G}$. In this work, we experiment with a diverse modes of attacks to show that our attack method can be generalised to different set-ups:

- creating/removing an edge: we create perturbed graphs by flipping the connection of a small set of node pairs $\delta\mathbf{A} = \{\{u_i, v_i\}\}_{i=1}^{\Delta}$ of $\mathcal{G}$ following previous works [37, 7];
- rewiring or swapping edges: similar to [17], we select a triplet (u, v, s) where we either rewire the edge $(u \to v)$ to $(u \to s)$ (rewire), or exchange the edge weights $w(u, v)$ and $w(u, s)$ (swap);
- node injection: we create new nodes together with their attributes and connections in the graph.

The overall routine of our proposed GRABNEL is presented in Algorithm 1, and we now elaborate each of its key components.

Surrogate model The success of BO hinges upon the surrogate model choice. Specifically, such a surrogate model needs to 1) be flexible and expressive enough to locally learn the latent mapping from a perturbed graph $\mathcal{G}'$ to its attack loss $\mathcal{L}_{\text{attack}}(f_\theta(\mathcal{G}'), y)$ (note that this is different and generally simpler than learning $\mathcal{G}' \to y$, which is the goal of the classifier f_θ), 2) admit a probabilistic interpretation of uncertainty, yet also 3) be simple enough such that the said mapping can be learned with a small number of queries to f_θ to preserve sample efficiency. Furthermore, given the combinatorial nature of the graph search space, it also needs to 4) be capable of scaling to large graphs (e.g. in the order of 10^3 nodes or more) typical of common graph classification tasks with reasonable run-time efficiency. Additionally, given the fact that BO has been predominantly studied in the continuous domain which is significantly different from the present setup, the design of a appropriate surrogate is highly non-trivial. To handle this set of conflicting desiderata, we propose to first use a *Weisfeiler-Lehman (WL) feature extractor* to extracts a vector space representation of $\mathcal{G}$, followed by a *sparse Bayesian linear regression* which balances performance with efficiency and gives an probabilistic output.

With reference to Algorithm 1, given a perturbation graph $\mathcal{G}'$ as a proposed adversarial sample, the WL feature extractor first extracts a vector representation $\boldsymbol{\phi}(\mathcal{G}')$ in line with the WL subtree kernel procedure (but without the final kernel computation) [24]. For the case where the node features are discrete, let $x^0(v)$ be the initial node feature of node $v \in \mathcal{V}$, we iteratively aggregate and hash the features of v with its neighbours, $\{u_i\}_{i=1}^{\deg(v)}$, using the original WL procedure at all nodes to transform them into discrete labels:

$$x^{h+1}(v) = \text{hash}\Big(x^h(v), x^h(u_1), ..., x^h(u_{\deg(v)})\Big),\ \forall h \in \{0, 1, \ldots, H-1\}, \tag{3}$$

where H is the total number of WL iterations, a hyperparameter of the procedure. At each level h, we compute the feature vector $\boldsymbol{\phi}_h(\mathcal{G}') = [c(\mathcal{G}', \mathcal{X}_{h1}), ..., c(\mathcal{G}', \mathcal{X}_{h|\mathcal{X}_h|})]^\top$, where $\mathcal{X}_h$ is the set of distinct node features x^h that occur in all input graphs at the current level and $c(G', x)$ is the counting function that counts the number of times a particular node feature x appears in G'. For the case with continuous node features and/or weighted edges, we instead use the modified WL procedure proprosed in [28]:

$$x^{h+1}(v) = \frac{1}{2}\Big(x^h(v) + \frac{1}{\deg(v)} \sum_{i=1}^{\deg(v)} w(v, u_i) x^h(u_i)\Big),\ \forall h \in \{0, 1, \ldots, H-1\}, \tag{4}$$

and we simply have $\boldsymbol{\phi}_h(\mathcal{G}') = \text{vec}(X_h)$ where we vectorise the feature matrix of graph $\mathcal{G}'$ at level h. In both cases, at the end of H WL iterations we obtain the feature vector $\boldsymbol{\phi}(\mathcal{G}') = \text{concat}\Big(\boldsymbol{\phi}_1(\mathcal{G}'), ..., \boldsymbol{\phi}_H(\mathcal{G}')\Big)$ for each training graph in $[1, n_{\mathcal{G}'}]$ to form the feature matrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathcal{G}'_1), ...\boldsymbol{\phi}(\mathcal{G}'_{|n_{\mathcal{G}'}|})]^\top$ to be passed to the Bayesian regressor. The WL iterations capture both information related to individual nodes and topological information (via neighbourhood aggregation),

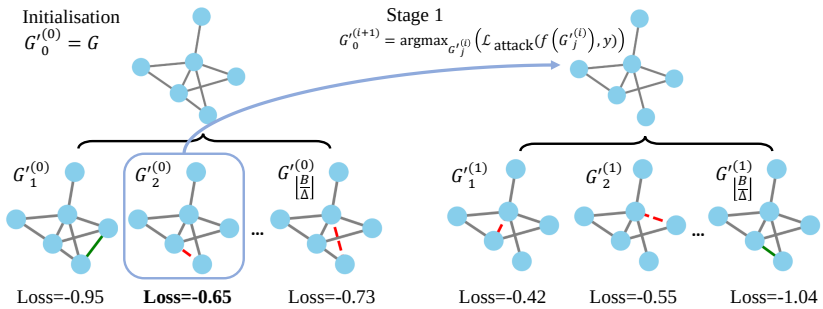


Figure 1: Sequential edge selection. At each stage the BO agent sequentially proposes candidate graphs with edge edit distance of 1 from the base graph $G_0'^{(i)}$ (which is the original unperturbed graph G at initialisation, or a perturbed graph that led to the largest increase in loss from the previous stage otherwise). This procedure repeats until either the attack succeeds (i.e. we find a graph G' with $\mathcal{L}_{\text{attack}}(f(G'), y) > 0$) or the maximum number of B queries to f_θ is exhausted.

and have been shown to have comparable distinguishing power to some GNN models [20], and hence the procedure is expressive. At the same time, the extraction process $\mathcal{G}' \to \boldsymbol{\phi}(\mathcal{G}')$ is also *unsupervised*, thereby avoiding the need for the surrogate to learn representation from the data to ensure good sample efficiency.

When H or $\mathcal{G}'$ (with many WL features) are large, the resulting feature matrix will likely be very high-dimensional, which would lead to high-variance regression coefficients $\boldsymbol{\alpha}$ being estimated if the number of input samples is comparatively few. To attain a good predictive performance in such a case, we employ Bayesian regression surrogate with the automatic relevance determination (ARD) prior to learn the mapping $\boldsymbol{\Phi} \to \mathcal{L}_{\text{attack}}(f_\theta(\mathcal{G}'), y)$, which regularises weights and encourages sparsity in $\boldsymbol{\alpha}$ [31]:

$$\mathcal{L}_{\text{attack}} | \boldsymbol{\Phi}, \boldsymbol{\alpha}, \sigma_n^2 \sim \mathcal{N}(\boldsymbol{\alpha}^\top \boldsymbol{\Phi}, \sigma_n^2 \boldsymbol{I}), \tag{5}$$

$$\boldsymbol{\alpha} | \boldsymbol{\lambda} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}),\ \text{diag}(\boldsymbol{\Lambda}) = \boldsymbol{\lambda}^{-1} = \{\lambda_1^{-1}, ..., \lambda_{\dim(\boldsymbol{\lambda})}^{-1}\}, \tag{6}$$

$$\lambda_i \sim \text{Gamma}(k, \theta)\ \forall i \in [1, \dim(\boldsymbol{\lambda})], \tag{7}$$

where $\boldsymbol{\Lambda}$ is the diagonal covariance matrix. To estimate $\boldsymbol{\alpha}$ and the noise variance σ_n^2, we optimise the model marginal log-likelihood. Overall, the WL routines scales as $\mathcal{O}(Hm)$, whereas training of Bayesian linear regression has a linear scaling w.r.t. the number of queries to the victim model; these ensure the surrogate is scalable to both larger graphs and/or a large number of graphs, both of which are commonly encountered in graph classification.

We note that alternative surrogate models for the BO include the GPWL surrogate proposed in [23] that directly uses a Gaussian Process (GP) model together with a WL kernel. Nonetheless, while GPs are theoretically more expressive (although we empirically show in App. B that in most of the cases their predictive performances are comparable), they are also much more expensive with a cubic scaling w.r.t the number of $\mathcal{G}'$. Furthermore, GPWL is designed specifically for the task of neural architecture search, which features small, directed graphs without edge weights and with discrete node features only; on the other hand, the surrogate in our paper covers a much wider scope of applications.

Sequential perturbation selection In the default structural perturbation setting, given an attack budget of Δ (i.e. we are allowed to flip up to Δ edges from $\mathcal{G}$), finding exactly the set of perturbations $\delta\mathbf{A}$ that leads to the largest increase in $\mathcal{L}_{\text{attack}}$ entails an combinatorial optimisation over $\binom{n^2}{\Delta}$ candidates. This is a huge search space that is difficult for the surrogate to learn meaningful patterns in a sample-efficient way even for modestly-sized graphs. To tackle this challenge, we adopt the strategy illustrated in Fig. 1: given the query budget B (i.e. the total number of times we are allowed to query f_θ for a given $\mathcal{G}$), we amortise B into Δ stages and focus on selecting *one* edge perturbation at each stage. While this strategy is greedy in the sense that it always commits the perturbation leading to the largest increase in loss at each stage, it is worth noting that we do *not* treat the previously modified edges differently, and the agent can, and does occasionally as we observe empirically, "correct" previous modifications by flipping edges back: this is possible due to the edge selection being permutation invariant. Another benefit of this strategy is that it can potentially make full use of

the entire attack budget Δ *while* remaining parsimonious w.r.t. the amount of perturbation introduced, as it only progresses to the next stage and modifies the $\mathcal{G}$ further when it fails to find a successful adversarial example in the current stage.

Optimisation of acquisition function At each BO iteration, there is need for the inner-loop optimisation of the acquisition function $\alpha(\cdot)$ to select the next point(s) to query f_θ (we use expected improvement (EI) [12] as the default acquisition function). While sample efficiency is typically not an issue in optimising α, the problem nonetheless entails the graph combinatorial space on which common tools like gradient-based optimisers cannot be used. A naïve strategy would be to generate a large number of randomly perturbed graphs, evaluate α on all of them, and choose the maximiser(s) to query f_θ next. While this strategy can be effective on modestly-sized $\mathcal{G}$ especially with our sequential selection strategy, it nevertheless discards any information we might have already learned about the search space.

Instead, we optimise α via the genetic algorithm (GA), which has recently been shown to be competitive in BO acquisition optimisation [5] and does not require a continuous search space. In our case, we adapt the GA proposed in [7] (note however that [7] directly uses it for adversarial attack whereas we only use it as a subroutine of the overall BO strategy) to our task and outline its ingredients below:

- *Initialisation:* While GA typically starts with random sampling in the search space to fill the initial population, in our case we are not totally ignorant about the search space as we might have already queried and observed f_θ with a few different perturbed graphs $\mathcal{G}'$. A possible smoothness assumption on the search space, for example, would be that if a $\mathcal{G}'$ with an edge (u, v) flipped from G led to a large $\mathcal{L}_{\text{attack}}$, then another $\mathcal{G}'$ with $(u, s), s \notin \{u, v\}$ flipped is more likely to do so too. To reflect this, we fill the initial population by *mutating* the top-k queried $\mathcal{G}'$s leading to the largest $\mathcal{L}_{\text{attack}}$ seen so far in the current stage, where for $\mathcal{G}'$ with (u, v) flipped from the base graph we 1) randomly choose an end node (u or v) and 2) change that node to another node in the graph except u or v such that the perturbed edges in all children shares one common end node with the parent.
- *Evolution*: After the initial population is built, we follow the standard evolution routine by evaluating the acquisition function value for each member as its *fitness*, selecting the top-k performing members as the breeding population and repeating the mutation procedure in initialisation for a fixed number of rounds. At termination, we simply query f_θ with the graph(s) seen so far (i.e. computing the loss in Fig. 1) with the largest acquisition function value(s) seen during the GA procedure.

3 Related Works

Adversarial attack on graph based models Recently there has been an increasing attention in the study of adversarial attacks in the context of graph neural networks [26, 11]. Applying adversarial attack methods from other domains to the graph setting is not straightforward, especially those based on gradient information, since the graph domain is inherently discrete. One of the earliest models, Nettack, attacks a GCN node classifier by optimising the attack loss of a surrogate model using a greedy algorithm [37]. Using a simple heuristic, DICE attacks node classifiers by adding edges between nodes of different classes and deleting edges connecting nodes of the same class [30]. Several reinforcement learning-based techniques [37, 17] have also been introduced for attacks on both node and graph classifiers . In [37] the authors use reinforcement learning to attack both node and graph classifiers. This method claims to be black-box. However, in the experiments for graph classification the surrogate model architecture is actually the same as the victim model. The approach was also not validated on real-world datasets. Another example is ReWatt, which also uses reinforcement learning to fool a graph classifier through rewiring [17]. Compared to both these methods, our method does not use the training data of the victim model and is much more query efficient. A white box optimisation strategy (alternating direction method of multipliers) is proposed in [10]. However, its specific to the GCN architecture, whereas our method is applicable to any graph-based model which outputs logits. Adversarial attacks on graph classifiers outside of the evasion setting have also been considered such as backdoor attacks [34, 32].

Adversarial attacks using BO BO as a means to find adversarial examples in the black-box evasion setting has been successfully proposed for classification models on tabular [27] as well as on image data [22, 35, 25, 21]. However, we address the problem for graph classification models, which work

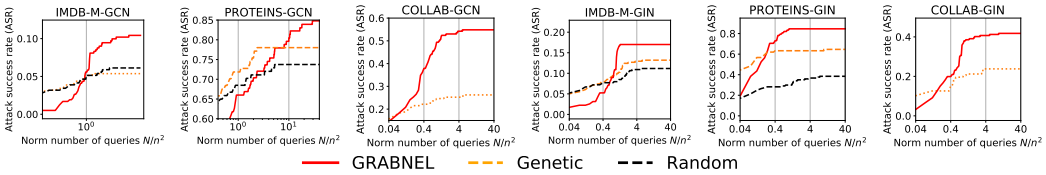


Figure 2: Attack success rate (ASR) against the number of queries to the victim models (normalised by the square of the number of nodes of each graph) in various datasets on GCN and GIN models. Note the x-axis is on log-scale. Lines and shades denote mean ± 1 sd across 3 random initialisations. It is evident that `grabnel` outperforms other attack methods considerably. `Random` and `Genetic` appear to converge faster in some tasks as they always use exploit full perturbation budget allocated, while `grabnel` only attempts a higher perturbation budget when attack fails in a lower one.

on structurally and topologically fundamentally different inputs. This implies several nontrivial challenges that require our method to go beyond the vanilla usage of BO. Firstly, the inputs cannot be readily represented as vectors like for tabular or image data. Secondly, the perturbations that we consider for such inputs are not defined on a continuous, but on a discrete domain. Lastly, the input dimensionality and the resulting number of queries necessary for finding successful adversarial examples in the graph setting goes beyond what has been addressed in the image classification setting.

4 Experiments

In this section, we validate the performance of the proposed method in a wide range of graph classification tasks with varying graph properties, including but not limited to the typical TU datasets considered in previous works [7, 17]. As a demonstration of the versatility of the proposed method, instead of considering a single mode of attack which is often impossible in real-life, we also select the attack mode specific to each task. All additional details, including the statistics of the datasets used and implementation details of the victim models and attack methods, are presented in App. C.

TU Datasets We first conduct experiments on three common TU datasets [19], namely (in ascending order of average graph sizes in the dataset) IMDB-M, PROTEINS and COLLAB. In all cases, we define the attack budget Δ in terms of the maximum *structural perturbation ratio* r defined in [4] where $\Delta \leq rN^2$. We similarly link the maximum numbers of queries B allowed for individual graphs to their sizes as $B = 50\Delta$, thereby giving larger graphs and thus potentially more difficult instances higher attack[3] and query budgets similar to the conventional image adversarial attack literature [22]. In this work, unless otherwise specified we set $r = 0.03$ for all experiments, and for comparison we consider a number of competitive baselines, including random search, genetic algorithm originally introduced in [7][4] and an additional simple gradient-based method which greedily adds or delete edges based on the magnitude computed input gradient similar to the gradient based method described in [7] (note that this method is *white-box* as access to parameter weights and gradients is required). To verify whether the proposed attack method can be used for a variety of classifier architectures we also consider two victim models, namely the Graph Convolutional Network (GCN) [14] and the Graph Isomorphism Network (GIN) [33]. We show the classification performance of both victim models before and after attacks using various methods in Table 1, and we show the attack success rate (ASR) against the (normalised) number of queries in Fig. 2. It is worth noting that in consistency with the image attack literature, we launch and consider attacks on the *graphs that were originally classified correctly*, and statistics, such as the ASR, are also computed on that basis.

The results generally show that the attack method is effective against both GCN and GIN models with GRABNEL typically leading to the largest degradation in victim predictions in all tasks, often performing on par or better than Gradient, a white-box method. Further, GRABNEL typically outperforms in a larger extent for the larger graphs (e.g. COLLAB) on which the benefit of the sequential selection of edge perturbation is more significant.

[3]Due to computational constraints, we cap the maximum number of queries to be 2×10^4 on each graph.

[4]Note that the original implementation of RL-S2V, the primary algorithm in [7], primarily focus on a S2V-based victim model[6]. We compare GRABNEL against it in the same dataset considered in [7] in App. D

Table 1: Validation accuracy of the victim models on the TU datasets before (*clean*) and after various attack methods. Results shown in mean $\pm$ 1 standard deviation across 3 trials.

	GCN [14]			GIN[33]		
	IMDB-M	PROTEINS	COLLAB	IMDB-M	PROTEINS	COLLAB
Clean	$50.53_{\pm 1.4}$	$71.73_{\pm 2.6}$	$79.73_{\pm 2.1}$	$48.85_{\pm 0.4}$	$70.53_{\pm 2.3}$	$80.80_{\pm 0.9}$
Random	$47.43_{\pm 1.2}$	$19.46_{\pm 1.7}$	$76.41_{\pm 6.2}$	$40.44_{\pm 2.5}$	$42.90_{\pm 2.2}$	$71.29_{\pm 0.9}$
Genetic [7]	$47.82_{\pm 1.5}$	$14.88_{\pm 1.7}$	$58.61_{\pm 7.9}$	$39.68_{\pm 3.1}$	$23.25_{\pm 5.3}$	$61.68_{\pm 2.5}$
Gradient-based†	$\mathbf{39.31_{\pm 2.2}}$	$50.60_{\pm 4.5}$	$36.67_{\pm 1.2}$	$\mathbf{37.56_{\pm 2.2}}$	$11.90_{\pm 4.4}$	$\mathbf{54.00_{\pm 2.9}}$
GRABNEL (ours)	$45.23_{\pm 0.2}$	$\mathbf{10.82_{\pm 2.5}}$	$\mathbf{35.38_{\pm 9.3}}$	$38.22_{\pm 3.9}$	$\mathbf{10.72_{\pm 7.1}}$	$57.33_{\pm 4.7}$

†: White-box method

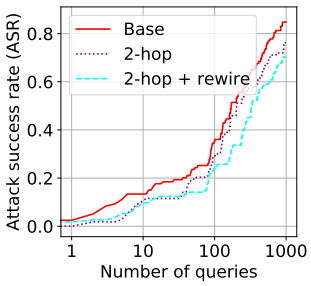


Figure 3: ASR vs # queries with constraints on a GCN model trained on PROTEINS dataset.

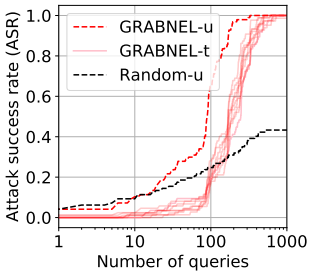


Figure 4: ASR vs # queries on a ChebyGIN trained on MNIST-75sp on targeted and untargeted attack setups.

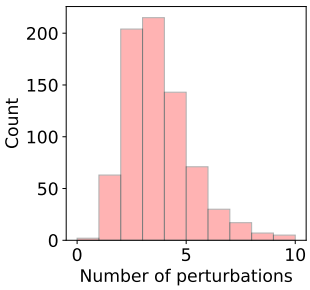


Figure 5: Histogram of number of edges swapped in successfully attacked samples of GRABNEL-t.

As discussed, in real life, adversarial agents might encounter additional constraints other than the number of queries to the victim model or the amount of perturbation introduced. To demonstrate that our framework can handle such constraints, we further carry out attacks on victim models using identical protocols as above but with a variety of additional constraints considered in several previous works. Specifically, the scenarios considered, in the ascending order of restrictiveness, are:

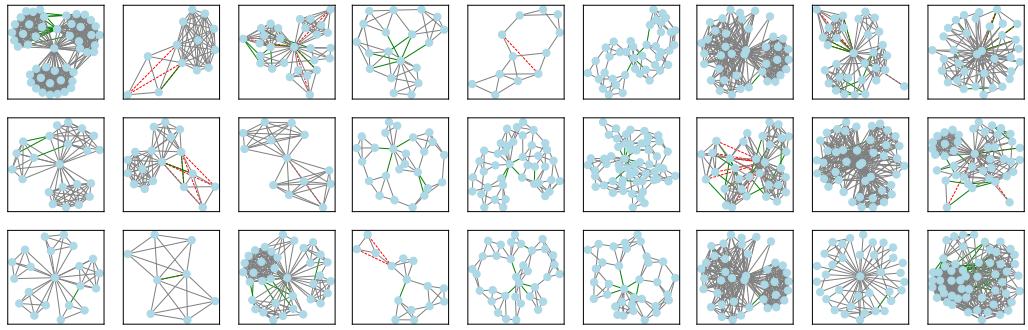
- `Base`: The base scenario is identical to the setup in Table 1 and Fig 2;
- `2-hop`: Edge addition between nodes (u, v) is only permitted if v is within 2-hop distance of u;
- `2-hop+rewire` [17]: Instead of flipping edges, the adversarial agent is only allowed to rewire from nodes (u, v) (where an edge exists) to (u, w) (where no edge currently exists). Node w must be within 2-hop distance of u;

We test on the PROTEINS dataset, and show the results in Fig. 3, where it is evident that while additional constraints unsurprisingly lead to (slightly) lower attack success rates, the performance of GRABNEL remains relatively robust in all scenarios considered. In fact, as we elaborate in Sec. 6, we find the phenomenon of attacked edges remaining relatively clustered within a relatively small neighbourhood is a general pattern in adversarial examples of many tasks. This implies that the 2-hop condition, which constrains the spatial relations of the adversarial edges, might already hold even without explicit specification, thereby explaining the marginal difference between the base and the 2-hop constrained cases in Fig. 3.

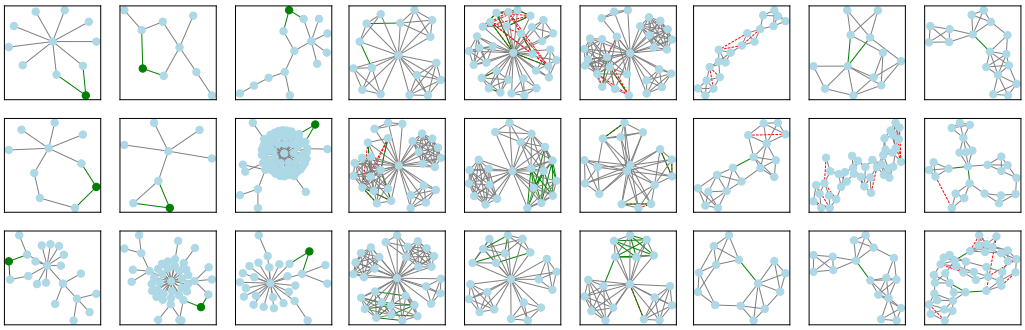
Image Classification Moving beyond the typical "edge flipping" setup on which existing research has been mainly focused, we now consider a very different setup involving attacks on the MNIST-75sp dataset [15] consisting weighted graphs with continuous attributes that are derived from MNIST [16]. The dataset is generated by first partitioning individual MNIST image into around 75 *superpixels* with SLIC [1, 8], which form the nodes of the graphs (with average superpixel intensity as node attributes). The pairwise distances between the superpixels, which form the edge weights, are then computed. We use the open-sourced, pre-trained ChebyGIN with attention model[5] released by the original authors [15] (with an average validation classification accuracy of around 95%) as the victim model.

Given that the edge values are no longer binary, simply flipping the edges (equivalent to setting edge weights to 0 and 1) is no longer appropriate. To generalise the sparse perturbation setup and inspired

[5]See `https://github.com/bknyaz/graph_attention_pool`.



(a) IMDB-M (GCN) (b) PROTEINS (GCN) (c) COLLAB (GCN)



(d) Twitter fake news (e) IMDB-M (GIN) (f) PROTEINS (GIN)

Figure 6: Adversarial examples found by the proposed method. Red edges denote deleted edges from the original samples and green edges indicate those added. In Twitter fake news detection task, green nodes/edges denote the injected nodes and their connections to the existing graphs.

by edge rewiring studied by previous literature, we instead adopt an attack mode via *swapping edges*: each perturbation can be defined by 3 end nodes (u, v, s) where edge weights w_{uv} is swapped with edge weight $w(u, s)$. We show the results in Fig. 4: `GRABNEL-u` and `Random-u` denotes the GRABNEL and random search under the *untargeted* attack, respectively, whereas `GRABNEL-t` denotes GRABNEL under the *targeted attack* with each line denoting 1 of the 9 possible target classes in MNIST. We find that GRABNEL is surprisingly effective in attacking this victim model, almost completely degrading the victim (Fig. 4) with very few swapping operations (Fig. 5) even in the more challenging targeted setup. This seems to suggest that, at least for the data considered, the victim model is very brittle towards carefully crafted edge swapping, with its predictive power seemingly hinged upon a very small number of key edges. We believe a thorough analysis of this is of independent interest, which we defer to a future work.

Fake news detection As a final experiment, we consider a real-life task of attacking a GCN-based fake news detector trained on a labelled dataset in [29]. Each discussion cascade (i.e. a chain of tweets, replies and retweets) is represented as an undirected graph, where each node represents a Twitter account (with node features being the key properties of the account such as age and number of followers/followees; see App. E for details) and each edge represents a reply/retweet. As a reflection of what a real-life adversary may and may not do, we note that modifying the connections or properties of the existing nodes, which correspond to modifying existing accounts and tweets, is considered impractical and prohibited. Instead, we consider a *node injection* attack mode (i.e. creating new malicious nodes and connect them to existing ones): injecting nodes is equivalent to creating new Twitter accounts and connecting them to the rest of the graph is equivalent to retweeting/replying existing accounts. We limit the maximum number of injected nodes to be $0.05N$ and the maximum number of new edges that may be created per each new node is set to the average number of edges an existing node

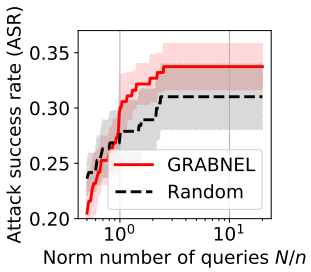


Figure 7: ASR vs. #queries (normalised by the number of nodes, since the attack involves node injection) on the Twitter dataset.

has – in this context, this limits the number of re-tweets and replies the new accounts may have to avoid easy detection. For the injected node, we initialise its node features in a way that reflects the characteristics of a new Twitter user (we outline the detailed way to do so in App. E). We show the result in Fig. 7, where GRABNEL is capable of reducing the effectiveness of a GCN-based fake news classifier by a third. In this case Random search also performs reasonably well, as the discussion cascade is typically small, allowing adversarial examples to be exhaustively searched.

Ablation Studies GRABNEL benefits from a number of key features, including but not limited to the Graph BO formulation, surrogate model choice and sequential perturbation selection. We investigate the relative contributions of them to the final performance via ablation studies in App. F.

5 Attack Analysis

Having established the effectiveness of our method, in this section we provide a qualitative analysis on the common interpretable patterns behind the adversarial samples found, which provides further insights into the robustness of graph classification models against structural attacks. We believe such analysis is especially valuable, as it may facilitate the development of even more effective attack methods, and may provide insights that could be useful for identification of real-life vulnerabilities for more effective defence.

- *Adversarial edges tend to cluster closely together*: A common observation across many datasets and models is that the distribution of the adversarial edges (either removal or addition) in a graph is highly uneven, with many adversarial edges often sharing common end-nodes or having small spatial distance to each other. This is empirically consistent with recent theoretical findings on the stability of spectral graph filters in [13]. From an attacker point of view, this may provide a "prior" on the attack to constrain the search space, as the regions around existing perturbations should be exploited more. As a proof of concept, we test the practical possibility of leveraging this to practically enhance attack performance and we show the results in App. G.
- *Adversarial edges often attempt to destroy or modify community structures*: for example, the original graphs in the IMDB-M dataset can be seen to have community structure. When the GCN model is attacked, the attack tends to flip the edges *between* the communities, and thereby destroying the structure by either merging communities or deleting edges within a cluster. On the other hand, the GIN examples tend to strengthen the community structures by adding edges within clusters and deleting edges between them. With similar observations also present in, for example, PROTEINS dataset, this may suggest that the models may be fragile to modification of the community structure.
- *Beware the low-degree nodes!* While low-degree nodes may be deemed less important in terms of degree centrality measures, we also find victim models might be particularly vulnerable to manipulations on such nodes. Most prominently, in the Twitter fake news example, the malicious nodes injected almost never opt to connect directly to the most central node (which is the original tweet) but instead wire to a peripheral node. This finding also corroborates the theoretical argument in [13] which show that spectral graph filters are more robust towards edge flipping involving *high-degree* nodes than otherwise.

6 Conclusion

This work proposes a novel and flexible black-box method to attack graph classifiers using Bayesian optimisation. We demonstrate the effectiveness and query efficiency of the method empirically. Unlike many existing works, we qualitatively analyse the adversarial examples generated. We believe such analysis is important to the understanding of adversarial robustness of graph-based learning models. Nevertheless, we would like to point out that a potential negative social impact of our work is bad actors using our method to attack real-world systems such as a fake new detection system on social media platforms. In future work, we will analyse adversarial patterns quantitatively and consider their effectiveness as a prior to guide the edge selection step of our method. A drawback of our current work is that it is specific to graph classification models. We believe it is possible to adapt our method to attack node classification models by suitably modifying the loss function. We leave this for future work.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[3] 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.

[4] Jinyin Chen, Dunjie Zhang, Zhaoyan Ming, and Kejie Huang. Graphattacker: A general multi-task graphattack framework. *arXiv preprint arXiv:2101.06855*, 2021.

[5] Alexander I Cowen-Rivers, Wenlong Lyu, Zhi Wang, Rasul Tutunov, Hao Jianye, Jun Wang, and Haitham Bou Ammar. Hebo: Heteroscedastic evolutionary bayesian optimisation. *arXiv preprint arXiv:2012.03826*, 2020.

[6] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711. PMLR, 2016.

[7] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018.

[8] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv*, 2020.

[9] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[10] Hongwei Jin, Zhan Shi, Venkata Jaya Shankar Ashish Peruri, and Xinhua Zhang. Certified robustness of graph convolution networks for graph classification under topological attacks. *Advances in Neural Information Processing Systems*, 33, 2020.

[11] Wei Jin, Yaxin Li, Han Xu, Yiqi Wang, and Jiliang Tang. Adversarial attacks and defenses on graphs: A review and empirical study. *arXiv preprint arXiv:2003.00653*, 2020.

[12] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[13] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. Interpretable stability bounds for spectral graph filters, 2021.

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

[15] Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. Understanding attention and generalization in graph neural networks. *arXiv*, (NeurIPS), 2019.

[16] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[17] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. Attacking graph convolutional networks via rewiring. *arXiv preprint arXiv:1906.03750*, 2019.

[18] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

[19] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[20] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[21] Luis Munoz-González. *Bayesian Optimization for Black-Box Evasion of Machine Learning Systems*. PhD thesis, Imperial College London, 2017.

[22] Binxin Ru, Adam Cobb, Arno Blaas, and Yarin Gal. Bayesopt adversarial attack. In *International Conference on Learning Representations*, 2019.

[23] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Neural architecture search using bayesian optimisation with weisfeiler-lehman kernel. *arXiv preprint arXiv:2006.07556*, 2020.

[24] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[25] Satya Narayan Shukla, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Black-box adversarial attacks with bayesian optimization. *arXiv preprint arXiv:1909.13857*, 2019.

[26] Lichao Sun, Yingtong Dou, Carl Yang, Ji Wang, Philip S Yu, Lifang He, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

[27] Fnu Suya, Yuan Tian, David Evans, and Paolo Papotti. Query-limited black-box attacks to classifiers. *NIPS Workshop*, 2017.

[28] Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. Wasserstein weisfeiler-lehman graph kernels. *arXiv preprint arXiv:1906.01277*, 2019.

[29] Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.

[30] Marcin Waniek, Tomasz P Michalak, Michael J Wooldridge, and Talal Rahwan. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139–147, 2018.

[31] David P Wipf, Srikantan S Nagarajan, J Platt, D Koller, and Y Singer. A new view of automatic relevance determination. In *NIPS*, pages 1625–1632, 2007.

[32] Jing Xu, Stjepan Picek, et al. Explainability-based backdoor attacks against graph neural networks. *arXiv preprint arXiv:2104.03674*, 2021.

[33] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[34] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Backdoor attacks to graph neural networks. *arXiv preprint arXiv:2006.11165*, 2020.

[35] Pu Zhao, Sijia Liu, Pin-Yu Chen, Nghia Hoang, Kaidi Xu, Bhavya Kailkhura, and Xue Lin. On the design of black-box adversarial examples by leveraging gradient-free optimization and operator splitting method. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 121–130, 2019.

[36] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[37] 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and the Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report error bars on all the experiments involving multiple trials in Section 4
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See the Appendix
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]