UnboundAttack : Generating Unbounded Adversarial Attacks to Graph Neural Networks

Sofiane Ennadir[1], Amr Alkhatib[1] Giannis Nikolentzos[2], Michalis Vazirgiannis[1,2], and Henrik Boström[1]

[1] EECS, KTH Royal Institute of Technology, Sweden
[2] LIX, Ecole Polytechnique, Paris

Abstract. Graph Neural Networks (GNNs) have demonstrated state-of-the-art performance in various graph representation learning tasks. Recently, studies revealed their vulnerability to adversarial attacks. While the available attack strategies are based on applying perturbations on existing graphs within a specific budget, proposed defense mechanisms successfully guard against this type of attack. This paper proposes a new perspective founded on unrestricted adversarial examples. We propose to produce adversarial attacks by generating completely new data points instead of perturbing existing ones. We introduce a framework, so-called UnboundAttack, leveraging the advancements in graph generation to produce graphs preserving the semantics of the available training data while misleading the targeted classifier. Importantly, our method does not assume any knowledge about the underlying architecture. Finally, we validate the effectiveness of our proposed method in a realistic setting related to molecular graphs.

Keywords: Adversarial Attacks, Graph Neural Networks

1 Introduction

In recent years, Graph Neural Networks (GNNs) emerged as an effective approach to learning powerful graph representations. These neural network-based models, for instance Graph Convolution Networks (GCNs) [12], have shown to be highly effective in a number of graph-based applications such as drug design [11]. However, recent literature has shown that these architectures can be attacked by injecting small perturbations into the input [2, 28]. These attacks, referred to as adversarial attacks in the literature, are highly critical, and this vulnerability has raised tremendous concerns about applying them in safety-critical applications such as financial and healthcare applications. For example, a malicious user could exploit this limitations by adding some inaccurate information to social networks. As a result, several studies focus on developing methods to mitigate the possible perturbation effects in parallel to these attacks. The proposed methods include adversarial training [6], enhancing the robustness of an input GNN through edge pruning [25], and recently proposing robustness certificates [16].

The currently available attacks are mainly based to applying small perturbations on either the structure or the node features of the graph [26, 23]. Given that

most of the proposed defense strategies enhance the robustness of the classifiers to small perturbations [10], they have shown some success in detecting these attacks and in limiting their effect. Moreover, most existing approaches formulate the problem of generating adversarial attacks as a search or constrained optimization problem. While the available constrained optimization tools are easily applicable in continuous input domains (i. e., images), adapting them to discrete domains such as graphs represents a significant challenge. Furthermore, in contrast to images, changing the graph structure by adding/deleting an edge may be infeasible and easily detectable in many settings. For instance, given a molecular graph where the edges represent chemical bonds, by deleting/adding an edge, the emerging graph may not represent a realistic molecule anymore.

To tackle the aforementioned limitations, in this paper, we introduce UnboundAttack, a more general and realistic attack mechanism which creates new adversarial examples from scratch instead of just applying perturbations to an input graph. The approach capitalizes on recent advancements in the field of Generative Adversarial Networks (GANs) to generate a set of legitimate graphs that share similar properties with the input graphs. These properties include degree distribution, diameter and subgraph structures among others. This approach of producing artificially generated graphs that do not emerge directly from input samples and which can mislead a targeted victim model is known as unbounded adversarial attacks. The term "unbounded" in this setting refers to the idea that these attacks are not directly linked to a specific existing graph but rather to a more general view of the dataset to be attacked. We validate in an experimental setting that these attacks can actually mislead the victim classifier but not some oracle function, thus presenting a major threat for real-world applications. The proposed framework is general and can operate on top of any GNN. Our main contributions are summarized as follows:

- We propose UnboundAttack, a generative framework for crafting from scratch adversarial attacks to pretrained GNNs. The proposed framework assumes no knowledge about the underlying architecture of the attacked model and may be applied to an ensemble of available models.
- We designed a realistic experimental setting using molecular data in which our model is evaluated and we show its effectiveness and ability to generate realistic and relevant attacks.

2 Related Work

Given the discrete nature of graphs, applying attack methods from other domains is very challenging. Similarly to the image domain attacks, most available methods formulate the task as a search problem. The objective of the task is to find the closest adversarial perturbation to a given input data point. This approach has led to several proposed attack strategies. For example, Nettack [26] introduced a targeted attack on both the graph structure and nodes features based on a greedy optimization algorithm of an attack loss to a surrogate model. In addition, [27] formulate the problem as a bi-level optimization task and leverages

meta-gradients to solve it. [23] expanded this work by proposing a black-box gradient attack algorithm to overcome several limitations of the original work. From another perspective, [3] propose to use Reinforcement Learning to solve the search problem and hence generate adversarial attacks. In the same context, the work [18] proposed to inject fake nodes into the graph and leveraged Reinforcement Learning to manipulate the labels and links of the injected nodes without changing the connectivity and other metrics between existing node. While the majority of the work is focusing on node classification, very few methods were proposed for the graph classification task. For instance, [19] proposed a new optimization-based approach to tackle the adversarial attack in a black-box setting. Moreover, [14] formulated the adversarial attack problem as an optimization problem and proposed an efficient solution based on a searching algorithm and query-efficient gradient. Finally, [24] designed an attack strategy based on learning a scoring function to rank nodes based on the attacker's expectation.

Formulating adversarial attacks as an optimal search problem has important limitations. For example, the search/optimization process should be performed for each attack input, which generally leads to a limited set of non-diverse perturbations. The attacks are therefore easily detectable by defense mechanisms such edge pruning [25]. In this perspective, unconstrained optimization approaches have lately been proposed to tackle these limitations. While the unconstrained approach has been investigated in other domains, such as images, to our knowledge, it has not been studied for graphs. For instance, [17] proposed synthesizing unrestricted adversarial images entirely from scratch using a conditional generative model and [1] proposed to semantically manipulate images to attack models.

3 Preliminaries

Before continuing with our contribution, we begin by introducing the graph classification problem and some key notation.

3.1 Graph Neural Networks

Let $G = (V, E)$ be a graph where V is its set of vertices and E its set of edges. We will denote by $n = |V|$ and $m = |E|$ the number of vertices and number of edges, respectively. Let $\mathcal{N}(v)$ denote the set of neighbors of a node $v \in V$, i. e., $\mathcal{N}(v) = \{u \colon (v, u) \in E\}$. The degree of a node is equal to its number of neighbors, i. e., equal to $|\mathcal{N}(v)|$ for a node $v \in V$. A graph is commonly represented by its adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ where the (i, j)-th element of this matrix is equal to the weight of the edge between the i-th and j-th node of the graph and a weight of 0 in case the edge does not exist. In some settings, the nodes of a graph might be annotated with feature vectors. We use $\mathbf{X} \in \mathbb{R}^{n \times K}$ to denote the node features where K is the feature dimensionality.
A GNN model consists of a series of neighborhood aggregation layers which use the graph structure and the nodes' feature vectors from the previous layer to generate new representations for the nodes. Specifically, GNNs update nodes'

feature vectors by aggregating local neighborhood information. Suppose we have a GNN model that contains T neighborhood aggregation layers. Let also $\mathbf{h}_v^{(0)}$ denote the initial feature vector of node v, i. e., the row of matrix $\mathbf{X}$ that corresponds to node v. At each iteration ($t > 0$), the hidden state $\mathbf{h}_v^{(t)}$ of a node v is updated as follows:

$$\mathbf{a}_v^{(t)} = \text{AGGREGATE}^{(t)}\Big(\big\{\mathbf{h}_u^{(t-1)} : u \in \mathcal{N}(v)\big\}\Big); \mathbf{h}_v^{(t)} = \text{COMBINE}^{(t)}\Big(\mathbf{h}_v^{(t-1)}, \mathbf{a}_v^{(t)}\Big) \tag{1}$$

where AGGREGATE is a permutation invariant function that maps the feature vectors of the neighbors of a node v to an aggregated vector. This aggregated vector is passed along with the previous representation of v (i. e., $\mathbf{h}_v^{(t-1)}$) to the COMBINE function which combines those two vectors and produces the new representation of v. After T iterations of neighborhood aggregation, to produce a graph-level representation, GNNs apply a permutation invariant readout function, e. g., the sum or mean operator, to nodes feature vectors as follows:

$$\mathbf{h}_G = \text{READOUT}\Big(\big\{\mathbf{h}_v^{(T)} : v \in V\big\}\Big) \tag{2}$$

3.2 Adversarial attacks

Given a classifier f, an input data point $G \in \mathcal{G}$ and its corresponding label $y \in \mathcal{Y}$ where $f(G) = y$, the goal of an adversarial attack is to produce a perturbed graph $\tilde{G}$ slightly different from the original graph with its predicted class being different from the predicted class of G. This could be formulated as finding a $\tilde{G}$ with $f(\tilde{G}) = \tilde{y} \neq y$ subject to $d(G, \tilde{G}) < \epsilon$ with d being some distance function between the original and perturbed graphs. For instance, d can be a matrix norm of the difference of the aligned adjacency matrices $|| \mathbf{A} - \tilde{\mathbf{A}} ||$ (e. g., l_∞, l_2).

4 Proposed Method: UnboundAttack

4.1 Unbounded adversarial attacks

We begin by formally defining what in what follows will be referred to as unbounded adversarial attacks. Given the set of graphs under consideration $\mathcal{G}$, let $o : \mathcal{O} \subset \mathcal{G} \rightarrow \{1, 2, .., K\}$ be the oracle from which the true labels of the graphs $\mathcal{Y}$ have been extracted. For instance, this oracle may be a human expert that uses domain knowledge to determine the category/class of a graph. As previously mentioned, a classifier denoted as $f : \mathcal{G} \rightarrow \mathcal{Y} = \{1, 2, .., Y\}$ (e. g., a GNN in our setting) is trained by minimizing a loss function (e. g., cross-entropy loss) in order to approach and estimate this oracle function. The model f is an estimator of the oracle o and therefore we assume that $f \neq o$. Given a graph $G \in \mathcal{G}$, an adversarial attack consists of finding a graph $\tilde{G} = (\tilde{V}, \tilde{E})$ slightly different from the original graph with its predicted class being different from the predicted class of G. In practice, the main objective is to generate a graph $\tilde{G}$ that shares

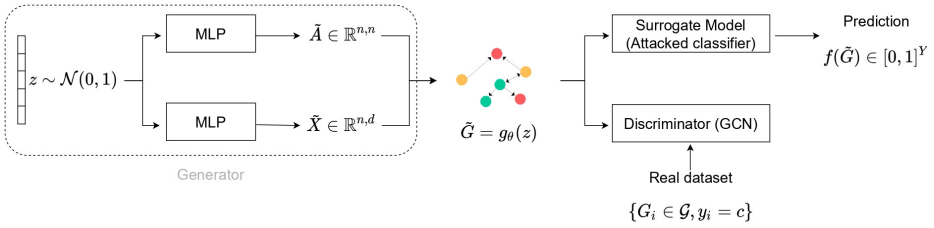

Fig. 1. Illustration of the proposed framework UnboundAttack for generating unbounded adversarial attacks. The framework consists of three main components : (1) A targeted GNN model; (2) A generator consisting of two MLPs taking a sampled vector as input. (3) A classifier distinguishing between generated and real graphs.

similar properties with the graphs of the training set (e. g., similar degree distribution, same motifs, etc.) as the graphs in the training set, and which can fool the classifier but not the oracle. We next define unbounded adversarial attacks.

Definition. (Unbounded adversarial attacks) *Given a small constant* $\epsilon > 0$ *and a graph comparison metric* d*, we define an unbounded adversarial example as a generated graph* $\tilde{G}$ *such as* $f(\tilde{G}) \neq o(\tilde{G})$ *and* $\forall G \in \mathcal{G}, d(G, \tilde{G}) < \epsilon$.

Notice that this definition can be seen as a generalization of the previous work on perturbation-based approaches since the generated graph needs to be similar to all graphs of the dataset. Note that d can be any function that measures distance of graphs, such as norm of the difference of their aligned adjacency matrices $\min_{\mathbf{P} \in \Pi} || \mathbf{A} - \mathbf{P}\tilde{\mathbf{A}}\mathbf{P}^\top ||$ where Π is the set of permutation matrices.

4.2 Architecture overview

As discussed, we seek to generate a graph with same characteristics as the graphs in the training set, for which the model's prediction differs from the class provided by the oracle. Our objective can be divided into two parts. The first part concerns generating realistic graphs with the same proprieties and semantics as our training set while the second part ensures reaching our adversarial aim.

Recurrent neural networks (RNNs) [22] or other likelihood-based generative models such as variational auto-encoders (VAEs) [13] may be suitable candidates to tackle this task. However, in practice, we choose to use a likelihood-free implicit generative approach; the generative adversarial network (GAN) [7], with a similar approach to prior work of [5]. We highlight that the primary contribution of our work is to provide a new adversarial perspective based on unbounded attacks. Consequently, while we have based our architecture on the GAN framework, other generative approaches could be used according to the targeted downstream task. The flow of our adversarial framework is summarized in Figure 1, we describe the different components in more details in what follows:

(1) ***Classifier.*** The victim classifier $f : \mathcal{G} \to [0,1]^Y$ is an instance of a GNN model following the general framework presented in subsection 3.1. We treat

the model as a gray-box. Thus, the model is supposed to be trained and fixed, and no assumptions are made about its internal architecture during the attack phase. Depending on the attack strategy, the attacker can chose to operate our framework as a white-box setting by directly using the victim classifier or as a gray-box setting by training a surrogate model.

*(2) **Generator.*** The generator g_θ learns to map a sampled D-dimensional vector from a normal distribution $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ into an adjacency matrix $\tilde{\mathbf{A}}$ and a feature matrix $\tilde{\mathbf{X}}$ representing a graph $\tilde{G}$. Given a previously chosen fixed graph size, we use two multi-layer perceptrons to process the input sampled vector. The output from each MLP is post-processed using a discretization strategy.

*(3) **Discriminator.*** The discriminator d_ϕ learns to distinguish between the generated and the real graphs. The model receives a synthetic graph and has to classify whether it is sampled from the true data distribution or generated by the generator g_θ. It should be noted that the discriminator needs to be invariant to the ordering of the nodes, and thus we employ a GCN [12].

4.3 Training and loss function

Generation loss. The main objective of the training phase is to generate realistic graphs with the same semantics as those of the training set. The training consists, therefore, of learning a relevant discriminator and generator. At each training step, the discriminator enhances its ability to distinguish between real graphs and generated ones, while the generator improves its capacity to generate graphs that mislead the discriminator. The process can be seen as a game between two active players (generator/discriminator) and a third static player (victim classifier) that should converge into an equilibrium. This equilibrium game is regulated by the classical GAN min-max equation:

$$\min_{g_\theta} \max_{d_\phi} \quad \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log d_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})}[\log(1 - d_\phi(g_\theta(\mathbf{z})))] \tag{3}$$

To ensure stability during training, we employ the Wasserstein GAN + GP (WGAN-GP) [8] that uses gradient norm penalty to achieve Lipschitz continuity.

Adversarial loss. Given our objective of generating unbounded adversarial attacks, by choosing a target attack class c, our model is trained to generate graphs that would be classified to the true class from the oracle (i. e., $o(\tilde{G})$=c) and to some other class from the classifier (i. e., $f(\tilde{G}) \neq c$). This is mainly reflected in the generator modeling the conditional data distribution $P(\cdot \mid y = c)$. In practice, during training, the discriminator is only given the set of real graphs whose training labels are equal to c, which is defined as: $\mathcal{G}_c = \{G_i \mid G_i \in \mathcal{G}, y_i = c\}$. Furthermore, the output of the generator is evaluated at each iteration by querying the attacked classifier. We strengthen the adversarial ability of the model by including an additional term in the loss function of the generator:

$$\mathcal{L}(\theta) = \mathcal{L}_{WGAN} + \beta \mathcal{L}_{Adv} \tag{4}$$

where $\beta \in [0, 1]$ is a trade-off parameter reflecting our desire to produce valid graphs and mislead the classifier. The second term of the loss function leads the output of the generator to be different from the target class. Specifically, $\mathcal{L}_{Adv}$ may be a reward function taking the generated graph as input and attributing a value based on the prediction probability formulated as $\mathcal{L}_{Adv} : \tilde{\mathcal{G}}_\theta \to \mathbb{R}$. For a single generated graph, we apply the following penalization term:

$$\mathcal{L}_{Adv}(\theta; \mathbf{z}) = \text{ReLU}\big(0.5 - \max_{i \neq c}(f(g_\theta(\mathbf{z}))_i\big) \tag{5}$$

where $\mathbf{z}$ refers to the i-th vector sampled from the normal distribution and given to the generator. Additionally, the $\max_{i \neq c}(f(g_\theta(\mathbf{z}))_i$ refers the maximum component (different from the c-th one) of the predicted probabilities vector of predicted probabilities. At each training step, we evaluate all the graphs produced by the generator given the sampled vectors from the normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. By minimizing this quantity, we maximize the other classes' (different from c) probabilities, therefore reaching our adversarial target.

Attacks Generation. After several training epochs, we expect the min-max game between the three players (i. e., generator, discriminator, and victim classifier) to converge to an equilibrium. Furthermore, we consider that by achieving this equilibrium, the generator can produce realistic graphs that are both adversarial and preserve the training set's main properties. Therefore, we directly leverage this trained generator during the testing phase to produce a set of adversarial graphs without querying the victim classifier or the discriminator.

5 Experimental Evaluation

In this section, we investigate the ability of the UnboundAttack framework to produce adversarial examples in a realistic experimental setting. We first describe the experimental setup, and then report the results and provide examples of generated graphs. More specifically, we address two main points: **(Q1)** Validity of attacks and **(Q2)** Adversarial aspect of these attacks.

5.1 Experimental setting

While experimenting with classical adversarial mechanisms is straightforward, evaluating our proposed approach is related to finding an accessible oracle capable of providing the labels of the generated graphs. In this experiment, we focus on chemical compounds using metrics available in the open-source chemoinformatics Python package RDKit. These metrics serve as our oracle from which we can derive the labels. We used the QM9 dataset [15] containing small organic molecules. Each of these available molecules is represented by an undirected graph G where nodes represent atoms and two atoms are connected by an edge if an atomic bond exists between them. Each node and edge in the graph is annotated with an one-hot vector indicating the type of the atom and atomic bond. We choose the following chemical-related metrics to be used as an oracle:

Table 1. Classification accuracy (± standard deviation) of the victim models on the Qm9 dataset on clean and attacked models. The lower the accuracy the better.

Attack strategy	Metric 1 - LogP		Metric 2 - SaS	
	GCN	**GIN**	**GCN**	**GIN**
Clean	97.8 % ± 0.7 %	97.1 % ± 0.2 %	91.4 % ± 0.3 %	89.8 % ± 0.2 %
Random	67.3 % ± 5.7 %	64.7 % ± 4.7 %	62.3 % ± 6.3 %	65.8 % ± 4.2 %
Gradient-based (PGD)	53.2 % ± 1.6 %	54.8 % ± 3.2 %	47.6 % ± 1.7 %	53.1 % ± 1.9 %
GradArgmax	48.5 % ± 2.7 %	51.7 % ± 2.3 %	45.3 % ± 2.4 %	54.9 % ± 3.6 %
Projective Ranking	47.8 % ± 2.3 %	58.3% ± 1.4 %	49.0 % ± 3.1 %	54.7 % ± 1.0 %
UnboundAttack	**45.9 %** ± 2.1 %	**47.3 %** ± 2.9 %	**27.1 %** ± 5.4 %	**31.2 %** ± 4.3 %
Attack strategy	Metric 3 - Density		Metric 4 - Weight	
	GCN	**GIN**	**GCN**	**GIN**
Clean	83.9 % ± 0.5 %	80.2 % ± 0.9 %	97.5 % ± 0.2 %	95.6 % ± 0.1 %
Random	58.3 % ± 4.9 %	67.3 % ± 5.2 %	59.4 % ± 4.2 %	63.7 % ± 5.7 %
Gradient-based (PGD)	49.5 % ± 1.7 %	54.2 % ± 2.7 %	54.0 % ± 2.9 %	51.5 % ± 2.2 %
GradArgmax	46.7 % ± 3.2 %	52.9 % ± 4.6 %	52.5 % ± 6.1 %	53.7 % ± 1.4 %
Projective Ranking	47.8 % ± 2.5 %	55.2 % ± 2.4 %	45.0 % ± 3.3 %	58.8 % ± 1.3 %
UnboundAttack	**43.2 %** ± 8.3 %	**49.7 %** ± 7.1 %	**30.3 %** ± 6.8 %	**40.7 %** ± 9.8 %

- The logP or Octanol-water partition coefficient is the partition coefficient representing the magnitude of the ratio of the concentration in Octanol.
- The Synthetic Accessibility score is a metric reflecting molecules' ease of synthesis (synthetic accessibility).
- The average molecular weight of the molecule and the molecule's density

Using RDKit, a score for each of the available graphs is calculated for the above metrics. By using a threshold (mean value), we convert each problem into a binary classification task. We used two 3-layers Multilayer Perceptron (MLP) models for the generator and a GCN as our discriminator. We arranged the number of message passing layers and hidden dimensions to be different from the victim model (especially in the GCN case) to assume no knowledge about the underlying architecture of the victim classifier. For our experimental evaluation, we used a discretization strategy based on the Gumbel-Softmax [9]. We demonstrate the UnboundAttack framework on two popular GNNs: (1) GCN [12]; and (2) GIN [21]. We used the sum operator as the readout function for both our victim model and the discriminator to produce graph-level representations. Furthermore, we used the cross-entropy loss with the Adam optimizer. We compare the proposed approach to other available methods, which are mainly based on constrained optimization, with respect to the accuracy of the attacker using a separate test set. We performed each experiment 3 times in a 3-fold cross-validation setting to estimate each attack's generalization performance.

5.2 Performance analysis

We first compare the method against the model's initial accuracy on test set (clean) before the attack. We additionally compare the proposed method against

Table 2. Results from the MMD and other metrics (± standard deviation) of the generated samples on the QM9 dataset for LogP metric.

Metric	MMD Metric		Other metrics	
	Deg.	**Clus.**	**Novelty**	**Uniqueness**
Gradient-based (PGD)	0.06 ± 0.01	0.03 ± 0.01	74.6 ± 1.2	66.3 ± 0.9
GradArgmax	0.08 ± 0.02	0.05 ± 0.02	68.6 ± 0.8	62.4 ± 1.1
Projective ranking	0.11 ± 0.01	0.03 ± 0.01	83.5 ± 1.8	82.7 ± 2.3
UnboundedAttack	0.14 ± 0.02	0.05 ± 0.04	89.7 ± 0.4	88.4 ± 0.6

different adversarial attack methods. The first comparison is against a random search method based on randomly adding/deleting edges. The second attack, adapted from [20], is a white-box gradient-based method that either adds/deletes edges by approaching the adversarial attack as an optimization method. The approach aims to find a set of perturbations that minimizes an attack loss given a finite budget of edge perturbations. We consider a specific budget Δ representing the maximum possible magnitude of the perturbations for both approaches and we used the Proximal Gradient descent as an optimization tool. Since this method is highly dependent on the chosen step-size ϵ, we tried different parameters and we reported the best result for each metric. We additionally compared our method to GradArgmax [4] which is based on a gradient-greedy method to select the optimal edge. After identifying the edges, removing or adding the edge depends on the sign of the gradient. We finally evaluate our method against ProjectiveRanking [24]. We note that the attack assumes access to the embedding representations of all nodes from the targeted classifier. Similar to their implementation, we used a 2-layers MLP to serve as the scoring module. We should mention that once the training phase of our method is completed, we can generate an unlimited number of adversarial attacks. In order to make the assessment fair, we set the number of generated points to be similar to the cardinality of the test set of other methods. Furthermore, we validate the quality of our generated attacks using two key perspectives. The first perspective, related to evaluating the quality of the generated graphs, is based on different graph metrics. We use the Maximum Mean Discrepancy (MMD) measures, as presented in [22], using the RBF kernel and for both the degree and clustering coefficient distributions. The second perspective is related to the biochemical validity of the generated samples and we used a Novelty and Uniqueness scores (similar to [5]) computed through RDKit. The classification performance of the GCN and GIN target models for all the metrics using different methods is reported in Table 1 and real examples are provided in Figure 2.

The results demonstrate the effectiveness of the proposed UnboundAttack strategy. The approach achieves the best attack performance on all datasets and the difference in performance between the proposed approach and the other baselines is significant. In addition, the comparison metrics shows that our method is capable of generating both valid and unique graphs. The obtained results thus answer **Q1** and demonstrate our generator's ability to provide pertinent adver-

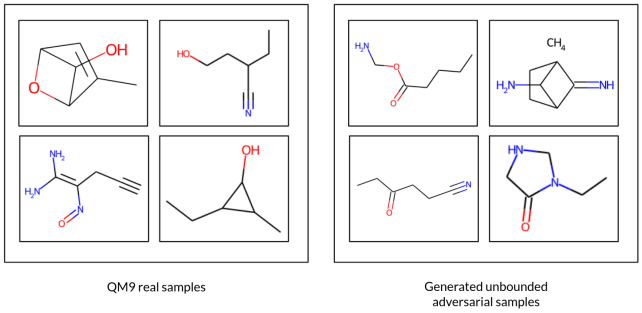

Fig. 2. Examples of graphs from the QM9 dataset *(left)*. Examples of generated attacks *(right)*. These examples have succeeded in misleading the classifier(i. e., $o(G) \neq f(G)$)
.

sarial attacks from scratch. While it may be argued that the computational cost of our method is higher, we should note that the validity of the output in terms of adversarial attacks is much more reliable. Moreover, in contrast with other methods, such as gradient-based methods, where a specific process is performed for each example, our training is only performed once. We would also like to mention our proposed method's ability to generate diverse valid adversarial attacks. Finally, contrary to the perturbation-based methods, our approach is not limited to the test set to be attacked but can generate a wide range of examples.

6 Conclusion

This work explores a new perspective on adversarial attacks on GNNs. Instead of performing perturbations on a graph by adding/removing edges or editing the nodes' feature vectors, we propose to learn a new graph from scratch using graph generative models. The produced graph has similar semantics to those of the graphs of the training set, and hence may be an effective tool to mislead a victim model. The proposed approach does not assume any knowledge about the architecture of the targeted model. Experiments show that the method performs better or comparable to other methods in degrading the performance of the victim model. This work can be extended to other graph setting such as node classification and edge classification. Furthermore, we anticipate that the proposed architecture may support the development of new defense strategies that could limit the potential negative impact of adversarial attacks, enhancing the ability to deploy GNNs in real practical settings.

Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

1. Bhattad, A., Chong, M.J., Liang, K., Li, B., Forsyth, D.A.: Unrestricted adversarial examples via semantic manipulation (2019). DOI 10.48550/ARXIV.1904.06347. URL https://arxiv.org/abs/1904.06347
2. Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., Song, L.: Adversarial attack on graph structured data (2018). DOI 10.48550/ARXIV.1806.02371. URL https://arxiv.org/abs/1806.02371
3. Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., Song, L.: Adversarial Attack on Graph Structured Data. In: Proceedings of the 35th International Conference on Machine Learning, pp. 1115–1124 (2018)
4. Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., Song, L.: Adversarial attack on graph structured data (2018). DOI 10.48550/ARXIV.1806.02371. URL https://arxiv.org/abs/1806.02371
5. De Cao, N., Kipf, T.: Molgan: An implicit generative model for small molecular graphs (2018). DOI 10.48550/ARXIV.1805.11973. URL https://arxiv.org/abs/1805.11973
6. Feng, F., He, X., Tang, J., Chua, T.S.: Graph adversarial training: Dynamically regularizing based on graph structure (2019). DOI 10.48550/ARXIV.1902.08226. URL https://arxiv.org/abs/1902.08226
7. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (2014). DOI 10.48550/ARXIV.1406.2661. URL https://arxiv.org/abs/1406.2661
8. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans (2017). DOI 10.48550/ARXIV.1704.00028. URL https://arxiv.org/abs/1704.00028
9. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax (2016). DOI 10.48550/ARXIV.1611.01144. URL https://arxiv.org/abs/1611.01144
10. Jin, W., Li, Y., Xu, H., Wang, Y., Ji, S., Aggarwal, C., Tang, J.: Adversarial attacks and defenses on graphs: A review, a tool and empirical studies (2020). DOI 10.48550/ARXIV.2003.00653. URL https://arxiv.org/abs/2003.00653
11. Kearnes, S., McCloskey, K., Berndl, M., Pande, V., Riley, P.: Molecular graph convolutions: moving beyond fingerprints. Journal of Computer-Aided Molecular Design **30**(8), 595–608 (2016)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016). DOI 10.48550/ARXIV.1609.02907. URL https://arxiv.org/abs/1609.02907
13. Kipf, T.N., Welling, M.: Variational graph auto-encoders (2016). DOI 10.48550/ARXIV.1611.07308. URL https://arxiv.org/abs/1611.07308
14. Mu, J., Wang, B., Li, Q., Sun, K., Xu, M., Liu, Z.: A hard label black-box adversarial attack against graph neural networks (2021). DOI 10.48550/ARXIV.2108.09513. URL https://arxiv.org/abs/2108.09513
15. Ramakrishnan, R., Dral, P.O., Rupp, M., von Lilienfeld, O.A.: Quantum chemistry structures and properties of 134 kilo molecules. Sci Data **1**, 140,022 (2014)
16. Schuchardt, J., Bojchevski, A., Gasteiger, J., Günnemann, S.: Collective robustness certificates: Exploiting interdependence in graph neural networks. In: International Conference on Learning Representations (2021). URL https://openreview.net/forum?id=ULQdiUTHe3y
17. Song, Y., Shu, R., Kushman, N., Ermon, S.: Constructing unrestricted adversarial examples with generative models (2018). DOI 10.48550/ARXIV.1805.07894. URL https://arxiv.org/abs/1805.07894

18. Sun, Y., Wang, S., Tang, X., Hsieh, T.Y., Honavar, V.: Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In: Proceedings of The Web Conference 2020, WWW '20, p. 673–683. Association for Computing Machinery, New York, NY, USA (2020). DOI 10.1145/3366423.3380149. URL https://doi.org/10.1145/3366423.3380149
19. Wan, X., Kenlay, H., Ru, B., Blaas, A., Osborne, M.A., Dong, X.: Adversarial attacks on graph classification via bayesian optimisation (2021). DOI 10.48550/ARXIV.2111.02842. URL https://arxiv.org/abs/2111.02842
20. Xu, K., Chen, H., Liu, S., Chen, P.Y., Weng, T.W., Hong, M., Lin, X.: Topology attack and defense for graph neural networks: An optimization perspective (2019). DOI 10.48550/ARXIV.1906.04214. URL https://arxiv.org/abs/1906.04214
21. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How Powerful are Graph Neural Networks? In: 7th International Conference on Learning Representations (2019)
22. You, J., Ying, R., Ren, X., Hamilton, W.L., Leskovec, J.: Graphrnn: Generating realistic graphs with deep auto-regressive models (2018). DOI 10.48550/ARXIV.1802.08773. URL https://arxiv.org/abs/1802.08773
23. Zhan, H., Pei, X.: Black-box Gradient Attack on Graph Neural Networks: Deeper Insights in Graph-based Attack and Defense. arXiv preprint arXiv:2104.15061 (2021)
24. Zhang, H., Wu, B., Yang, X., Zhou, C., Wang, S., Yuan, X., Pan, S.: Projective ranking: A transferable evasion attack method on graph neural networks. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21, p. 3617–3621. Association for Computing Machinery, New York, NY, USA (2021). DOI 10.1145/3459637.3482161. URL https://doi.org/10.1145/3459637.3482161
25. Zhang, X., Zitnik, M.: Gnnguard: Defending graph neural networks against adversarial attacks (2020). DOI 10.48550/ARXIV.2006.08149. URL https://arxiv.org/abs/2006.08149
26. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial Attacks on Neural Networks for Graph Data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2847–2856 (2018)
27. Zügner, D., Günnemann, S.: Adversarial attacks on graph neural networks via meta learning. In: 7th International Conference on Learning Representations (2019)
28. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2018)