Graph Attacks with Latent Variable Noise Modeling

Avishek Joey Bose *
McGill University, Mila

Andre Cianflone *
McGill University, Mila

William L. Hamilton *
McGill University, Mila

Abstract

Adversarial attacks on deep neural networks traditionally rely on a constrained optimization paradigm, where an optimization procedure is used to obtain a single adversarial perturbation for a given input example. In this work we frame the problem as learning a distribution of adversarial perturbation, enabling us to generate adversarial distributions given an unperturbed input. We apply this adversarial approach to the graph domain in the whitebox attack setting, achieving a new state-of-the-art. Finally, we demonstrate that our generative framework can efficiently generate a diverse set of attacks for a single given input, and is even capable of attacking *unseen* test instances in a zero-shot manner, exhibiting *attack generalization.*

1 Introduction

Adversarial attacks on deep learning models involve adding small, often imperceptible, perturbations to input data with the goal of forcing the model to make certain misclassifications [21]. There are a wide-variety of settings and assumptions under which adversarial strategies are developed, including the "whitebox" setting, where the model parameters are accessible to the attacker, and the more challenging "blackbox" setting, where the attacker only has access to the inputs and outputs of the target model [16, 17]. Despite this diversity, most existing approaches treat generating adversarial attacks as a constrained optimization or search problem [3]. The objective is to search for a specific adversarial perturbation based on a particular input datapoint, with the constraint that this perturbation be small. This paradigm has led to a number of highly successful attack strategies, particularly in the whitebox setting. Examples include the Fast Gradient Sign Method (FGSM) [7], L-BFGS [21], Jacobian-based Saliency Map Attack (JSMA) [18], DeepFool [15], Carlini-Wagner [3] and the PGD attack [14], to name a few.

However, this paradigm has important limitations. For example, while these attack strategies are easily applicable to continuous input domains (i.e., images), adapting them to new modalities, such as discrete textual data [12, 5, 6] or graph-structured data that is non-i.i.d. [4, 25], represents a significant challenge [23, 20]. In addition, a specific optimization must be performed for each attacked input, which generally only leads to a single or small set of non-diverse perturbations, which can make them easier to defend against. These approaches do not efficiently generalize to unseen examples, since they require a new round of optimization to be performed for each attacked input.

We propose GALVN (**G**raph-**A**ttacks with **L**atent **V**ariable **N**oise modeling); a generative framework for whitebox adversarial attacks on the graph domain. Our approach offers the following key benefits: **(1)** *Efficient generalization.* With our stochastic latent variable model we can efficiently construct adversarial examples and even generalize without any further optimization to unseen test examples. **(2)** *Diverse attacks.* We learn a diverse conditional distribution of adversarial examples, allowing us

*Correspondence to Authors: `{joey.bose@mail,andre.cianflone@mail,wlh@cs}.mcgill.ca` to *resample* after a failed attack. Our approach achieves a new state-of-the-art for attacking a Graph Neural Network using node features alone.

2 Background and Preliminaries

Given a classifier $f : \mathcal{X} \to \mathcal{Y}$, input datapoint $x \in \mathcal{X}$, and class label $y \in \mathcal{Y}$, where $f(x) = y$, the goal of an adversarial attack is to produce a perturbed datapoint $x' \in \mathcal{X}$ such that $f(x') = y' \neq y$, and where the distance $\Delta(x, x')$ between the original and perturbed points is sufficiently small.

Threat models. Adversarial attacks can be classified under different threat models, which impose different access and resource restrictions on the attacker [1]. In this work we consider the whitebox setting, where the attacker has full access to the model parameters and outputs. This setting is more permissive than the blackbox [16, 17] and semi-whitebox [24] settings, which we consider for test set attacks. Constrained optimization attacks in the whitebox setting are relatively mature and well-understood, making it a natural setting to contrast our alternative generative modelling-based attack strategy. In addition, we consider the more common setting of *untargeted* attacks, where the goal of the attacker is to change the original classification decision to any other class.[2]

Constrained optimization approach. In the standard constrained optimization framework, the goal is to find some minimal perturbation $\delta \in \mathcal{X}$ that minimizes $\Delta(x, x+\delta)$, subject to $f(x+\delta) = y'$ and $x + \delta \in \mathcal{X}$, where the last constraint is added to ensure that the resulting $x' = x + \delta$ is still a valid input (e.g., a valid image in normalized pixel space.) Rather than solving this non-convex optimization problem directly, the typical approach is to relax the constraints into the objective function and to employ standard gradient descent [7] or projected gradient descent, with the projection operator restricting x' such that $\Delta(x, x') \leq \epsilon$ and $x' \in \mathcal{X}$ [14].

Limitations of the constrained optimization approach. The constrained optimization formulation has been effectively deployed in many recent works, especially for images (see [1] for a survey). However, this approach has two key technical limitations: **(1)** Directly searching for an adversarial example x' or an adversarial perturbation $\delta \in \mathcal{X}$ can be computationally expensive if the input space $\mathcal{X}$ is very high-dimensional or discrete (e.g., for graphs). Moreover, adding the perturbation δ to x is only feasible if the input domain $\mathcal{X}$ is a field with a well-defined notion of addition, which is not the case for discrete domains such as graphs. **(2)** A distinct optimization procedure must be run separately for each attacked datapoint, generally yielding a single (or small set) of non-diverse perturbations per attacked datapoint. Together, these limitations make it non-trivial to generalize the constrained optimization approach beyond the image domain, and make it difficult to generate diverse attacks for unseen datapoints.

3 Proposed Approach

To address these limitations, we propose adversarial attacks in the latent variable modeling framework. Instead of searching over the original input space $\mathcal{X}$ to generate an adversarial example, we learn to generate perturbations within a low-dimensional, continuous latent space $\mathbb{R}^d$. We define an adversarial generator network, G, which specifies a conditional distribution $P(x'|x)$ over adversarial examples x' given an input datapoint x. The model is defined as a combination of four components (Figure 2): **(1)** A *probabilistic encoder network*, $q_\phi(z|x)$ that defines a conditional distribution over latent codes $z \in \mathbb{R}^d$ given an input $x_i \in \mathcal{X}$. The latent conditional distribution as $z \sim \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$, where $\mu_\phi, \sigma_\phi : \mathcal{X} \to \mathbb{R}^d$ are differentiable neural networks. **(2)** A *probabilistic decoder network*, $p_\theta(\delta|z)$ that maps a sampled latent code $z \in \mathbb{R}^d$ to a perturbation δ. **(3)** A *combination function* $c(\delta, x)$ that combines δ and x to output a perturbed example x'. **(4)** A *similarity function* Δ defined over $\mathcal{X}$, used to restrict the space of adversarial examples.

Adversarial generalization. Instead of performing a specific optimization for each attacked example, we can generate a distribution of adversarial examples on arbitrary input points, even points that were unseen during training. In Section 4.1, we demonstrate how resampling significantly improves attack generalization on a test set.

[2]Our framework could be extended to the targeted setting by modifying the loss function, as well as to blackbox attacks via blackbox gradient estimation [22, 8], but we leave these extensions to future work.

3.1 Training and loss function

Misclassification loss. We use a max-margin misclassification loss, since it has been shown to be robust to hyperparameter settings [3]. Using $s(x, y) \in \mathbb{R}$ to denote the (unnormalized) relative likelihood that the classifier f assigns to point $x \in \mathcal{X}$ belonging to class $y \in \mathcal{Y}$, we define the classification loss as:

$$\mathcal{L}_c = \frac{1}{N}\sum_{i}^{N} \max(s(x_i', y_i) - (s(x_i', y_i'))_{\max_{y_i' \neq y_i}}, 0),$$

where y_i is the correct class for the unperturbed point x_i.

Regularization. We penalize the model according to the similarity $\Delta(x, x')$ between the perturbed and unperturbed points. To avoid code collapse, the latent code is KL-regularized with a uniform Gaussian.

Overall objective and training. The overall objective is thus defined as follows:

$$\mathcal{L} = \mathcal{L}_c + \frac{1}{N}\sum_{i}^{N}(-\lambda \cdot \Delta(x_i, x_i') + D_{KL}(q_\phi(z|x_i)\,||\,\mathcal{N}(0,1)). \tag{1}$$

Objective (1) is minimized end-to-end with stochastic gradient descent. While the generative model resembles a variational auto-encoder (VAE) [10], our objective function is an adversarial loss in constrast to the usual cross-entropy objective. Moreoever, the reconstruction error in Equation (1) is given by an arbitrary similarity function Δ, with a hyperparameter λ that trades-off the adversarial misclassification objective from the magnitude of the perturbation by maximizing the similarity.

3.2 Implementation

We consider the challenging attack setting where the attack model can only make changes to node attributes and not the graph structure itself [25]—employing a graph convolution network (GCN) [11] as the encoder network and a multi-layer perceptron (MLP) as the decoder. Note, however, that adversarial attacks on graphs present unique complications compared to texts and images in that the training data is non-i.i.d., with the training points being a sub-sample of the nodes in a large graph. Thus, following Zugner et al. [25], we consider both *direct attacks*, which modify the features of the target nodes themselves as well as *influencer attacks*, which can only modify features in the neighborhood of a target node but not the node itself. Consequently, we define two sets of disjoint nodes: the attacker set $\mathcal{A}$, and the target set $\mathcal{T}$. For direct attacks $\mathcal{A} = \mathcal{T}$ and in this case the combination function is simply $c(\delta, x) = x + \delta$. For influencer attacks, only the embeddings of the $\mathcal{A}$ are modified and thus we use a binary mask in our combination function —i.e. $c(\delta, x) = x + b \cdot \delta$, where $b \in [0, 1]^N$. We use the l_2 norm as the similarity function.

4 Experiments

We investigate the application of our generative framework to produce adversarial examples against two standard node classification datasets: Cora and CiteSeer [13, 2, 19]. As points of comparison throughout our experiments, we consider NetAttack approach [25]—a powerful baseline that relies on a constrained optimization based approach to generate adversarial examples. We also experiment with a simplified version of GALVN that uses a deterministic autoencoder, rather than a variational approach (denoted GALVN-AE). We split the dataset into labeled 20% of which 10% is used for training and the remaining is used for validation. The remaining nodes are unlabeled nodes and are used for testing purposes. We attack a single-layer graph convolutional network (GCN) model [11] that is trained for 100 epochs with the Adam optimizer and a learning rate of $1e-2$. Following previous work, we consider both *direct* and *influencer* attacks (as discussed in Section 3.2); in the influencer setting we attack a node by changing the features of a random sample of 5 of its neighbors.[3]

[3]If there are fewer than 5 neighbors we randomly sample nodes from $\mathcal{A}$ until we have a total of 5

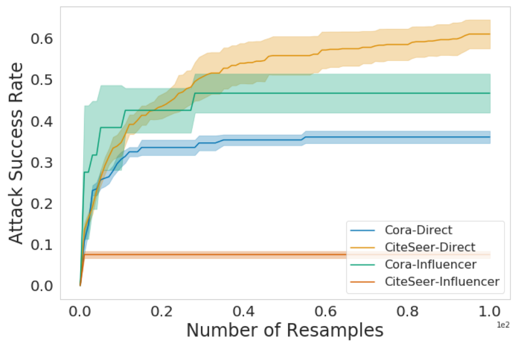

Figure 1: Attack Success Rate when resampling only failed adversarial examples in the test set.

	Cora		CiteSeer	
	Train	Test	Train	Test
GD	88 %	91 %	81%	82%
GAED	**100%**	**93%**	**100%**	**92%**
ZD	99%	-	99%	-
DI	**62%**	-	**54 %**	-
GAEI [25]	60%	-	52%	-
ZI [25]	33 %	-	38%	-

Table 1: Attack success rate on Graph Data. (GD) GALVN-Direct, (GAED) GALVN-AE-Direct, (ZD) Zugner-Direct, (DI) GALVN-Influencer, (GAEI) GALVN-AE-Influencer, (ZI) Zugner-Influencer. We used $\lambda = 0.01$ and results are averaged over 5 runs.

4.1 Results

Attack Strength. We find that GALVN achieves very strong results (Table 1) and our GALVN-AE baseline achieves a new state-of-the-art for *direct attacks* (when modifying node features only), outperforming the constrained optimization NetAttack approach [25] and achieving a perfect success rate on the training set (Table 1). Both GALVN-AE and GALVN also significantly outperform NetAttack in the *influencer attack* setting, with the latter seeing an absolute improvement of 29% and 16% on Cora and CiteSeer, respectively. Note that we do not compare against the graph attack framework of Dai et al. [4], since that work modifies the adjacency matrix of the graph, whereas we modify node features only.

Attack Generalization. We demonstrate the ability of GALVN to attack a test set of previously *unseen* instances, with results summarized under the Test columns in Tables 1. Since constrained optimization-based attack strategies cannot generalize to a test set of examples, we rely on our GALVN-AE approach as a strong baseline. We observe that GALVN has marginally better generalization ability for influencer attacks while GALVN-AE performs better by a similar margin in the easier direct attack setting.

Diversity of Attacks. GALVN exhibits comparable (or marginally worse) performance compared to our deterministic autoencoder (GALVN-AE) approach in terms of raw success rates. However, a key benefit of the full GALVN framework is that it has a stochastic latent state, which allows for resampling of latent codes to produce new adversarial examples. We empirically verify this phenomena by resampling failed test set examples up to a maximum of 100 resamples. As can be seen from Figure 1, GALVN can produce adversarial samples for clean inputs that were originally classified correctly, significantly boosting generalization performance. We resample test set instances for direct attacks and failed training set instances for influencer attacks but still without any further optimization. We achieve significant improvements of 36% and 61% for Cora and CiteSeer direct attacks, respectively, and 47% and 8% for Cora and CiteSeer influencer attacks, respectively.

5 Discussion and Conclusion

We present GALVN, a framework for constructing adversarial attacks on the graph domain. We successfully show that our trained model is capable of generating adversarial examples on unseen test examples in a new form of attack generalization. Further, we show that the generative nature of GALVN makes it capable of generating diverse sets of attacks for any given input, allowing the attacker to cheaply resample failed attacks in an another attempt to be adversarial. Future work will focus on extending GALVN to flexible posterior distributions, such as normalizing flows. Additionally, further studies are necessary to demonstrate how training a target model on these attacks increases its robustness

Acknowledgements

The authors would like to thank Maxime Wabartha, Ariella Smofsky, Lucas Caccia, Nadeem Ward, Yanshuai Cao, and Gavin Ding for helpful feedback on earlier drafts of this work. Andre Cianflone is supported by the Borealis AI fellowship, and acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number 411304954]. This research was funded in part by an academic grant from Microsoft Research, as well as a Canada CIFAR Chair in AI, held by Prof. Hamilton.

References

[1] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

[2] Indrajit Bhattacharya and Lise Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

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

[4] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*, 2018.

[5] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017.

[6] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE, 2018.

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

[8] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations (ICLR)*, 2018.

[9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[11] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[12] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.

[13] Qing Lu and Lise Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 496–503, 2003.

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[15] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016.

[16] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[17] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[18] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[19] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[20] Lichao Sun, Ji Wang, Philip S Yu, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

[21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[22] George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pages 2627–2636, 2017.

[23] Wenqi Wang, Benxiao Tang, Run Wang, Lina Wang, and Aoshuang Ye. A survey on adversarial attacks and defenses in text. *CoRR*, abs/1902.07285, 2019.

[24] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.

[25] 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. ACM, 2018.

A Implementation Details

In general we found it useful to spend multiple computation cycles on a particular batch before moving on to the next batch. We used an upper limit of 40 gradient computations per batch or untill all examples in the batch were misclassified and the δ was lower than some threshold. To tune our hyperparameters we used the same train/validation splits that were used to train the target classifier.

Attack Domain	Input Type	Encoder	Decoder	δ	Δ	$c(x, \delta)$
Graph-Direct	Node Emb	GCN	MLP	Node Emb	l_2	$x + \delta$
Graph-Influencer	Node Emb	GCN	MLP	Node Emb	l_2	$x + b \cdot \delta$

Table 2: Summary of the different components to applying our framework

Hyperparameters. We now detail exact configuration of hyperparameter settings used in all of our experiments.

Parameter	Description	Value
Generator Architecture	GCN Encoder and MLP Decoder	-
GCN hidden size	Number of features for 1-layer GCN	16
Attack Epochs	Number of Epochs for GALVNtraining	200
Latent size	AE and GALVN latent posterior size	50
Dropout Ratio	Used for Regularization	0.5
$\lvert\mathcal{T}\rvert$	Number of Target Nodes	40
Learning rate	Learning rate for G	0.01
Optimizer	GALVN optimizer	Adam [9]
λ	L2 regularization coefficient	0.01
Batch size	For training and testing	256
GPU	For training and testing	1 Nvidia 1080 Ti
Similarity function	Metric used for Δ	l_2-distance

Table 3: Hyperparameter and experimental details for the Graph domain

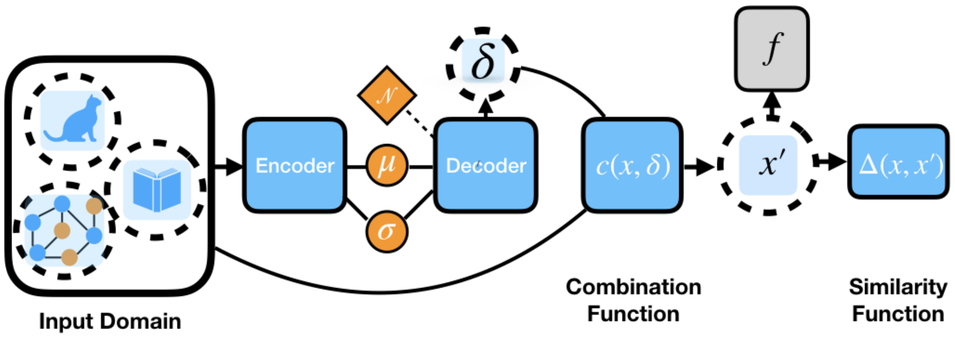

Figure 2: The main components of GALVN and the forward generation of an adversarial example.