Generalizable Adversarial Attacks Using Generative Models

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. Here, we instead view adversarial attacks as a generative modelling problem, with the goal of producing entire distributions of adversarial examples given an unperturbed input. We show that this generative perspective can be used to design a unified encoder-decoder framework, which is *domain-agnostic* in that the same framework can be employed to attack different domains with minimal modification. Across three diverse domains—images, text, and graphs—our approach generates whitebox attacks with success rates that are competitive with or superior to existing approaches, with a new state-of-the-art achieved in the graph domain. 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 [33]. There are a wide-variety of settings and assumptions under which adversarial strategies are developed (often termed the "threat model"). This includes the so-called "whitebox" setting, where the model parameters are fully accessible to the attacker, as well as the more challenging setting of "blackbox" attacks, where the attacker only has access to the inputs and outputs of the target model [24, 25].

Despite this diversity, a commonality in most existing approaches is the treatment of generating adversarial attacks as a constrained optimization or search problem [5]. In this constrained optimization paradigm, the objective is to search for a specific adversarial perturbation based on a particular input datapoint, with the constraint that this perturbation is not too large. The constrained optimization paradigm has led to a number of highly successful attack strategies, with whitebox attacks based on first-order gradient information being particularly ubiquitous and successful. Examples of successful strategies employing this approach include the classic Fast Gradient Sign Method (FGSM) [10], as well as more recent variants such as L-BFGS [33], Jacobian-based Saliency Map Attack (JSMA) [26], DeepFool [23], Carlini-Wagner [5] and the PGD attack [22], to name a few.

However, the constrained optimization paradigm has important limitations. For example, while constrained optimization-based attack strategies are easily applicable to continuous input domains (i.e., images), adapting them to new modalities, such as discrete textual data [18, 7, 8] or graph-structured data that is non-i.i.d. [6, 38], represents a significant challenge [35, 32]. In addition, in the constrained optimization approach, a specific optimization must be performed for each attacked input, which generally only leads to a single or small set of non-diverse perturbations. As a consequence,

[*] Correspondence to Authors: {joey.bose@mail,andre.cianflone@mail,wlh@cs}.mcgill.ca

Preprint. Under review.

constrained optimization approaches often fail to produce diverse attacks—which can make them easier to defend against—and these approaches do not efficiently generalize to unseen examples, since they require a new round of optimization to be performed for each attacked input.

Present work. We propose DAGAER (**D**omain-**A**gnostic **G**enerative **A**dversarial **E**xamples with **R**esampling; pronounced "dagger"), a unified generative framework for whitebox adversarial attacks, which is easily adaptable to different input domains and can efficiently generate diverse adversarial perturbations. DAGAER leverages advancements in deep generative modeling to learn an encoder-decoder based model with a stochastic latent variable that can be used to craft small perturbations.

Our approach offers three key benefits:

- **Domain-agnostic.** Our framework can be easily deployed in diverse domains (e.g., images, text, or graphs) by simply choosing an appropriate encoder, decoder, and similarity function.
- **Efficient generalization.** Employing a parametric model with a stochastic latent variable allows us to amortize the inference process when crafting adversarial examples. After training, we can efficiently construct these adversarial examples and even generalize without any further optimization to unseen test examples with only a single pass through the trained network.
- **Diverse attacks.** We learn a conditional distribution of adversarial examples, which can be efficiently sampled to produce diverse attacks or used to *resample* when an initial attack fails.

To demonstrate the benefits of DAGAER we implement variants to attack CNN-based image classifiers, LSTM-based text classifiers, and GNN-based node classification models. In the image domain, our framework achieves results that are competitive with constrained optimization-based approaches in terms of attack success rates, while providing more efficient generalization and diverse perturbations. In the text domain, we provide strong results on the IMDB benchmark, being the first work to scale an approach with state-of-the-art success rates to the full test set. Finally, in the graph domain, our approach achieves a new state-of-the-art for attacking a GNN 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 [24, 25] and semi-whitebox [36] 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 [10] or projected gradient descent, with the projection operator restricting x' such that $\Delta(x, x') \leq \epsilon$ and $x' \in \mathcal{X}$ [22].

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 text or graphs). Moreover, adding the perturbation δ to x is only feasible if the input domain $\mathcal{X}$ is a

[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 [34, 11], but we leave these extensions to future work.

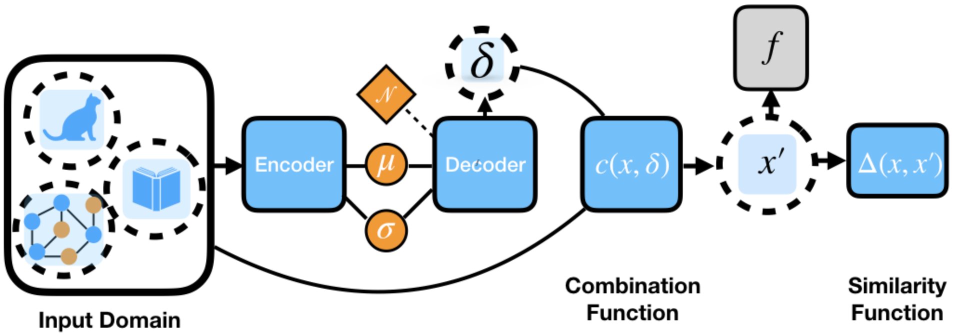

Figure 1: The main components of DAGAER and the forward generation of an adversarial example.

field with a well-defined notion of addition, which is not the case for discrete domains such as text.

2. A distinct optimization procedure must be run separately for each attacked datapoint, and without substantial modifications, this optimization will only yield 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 the limitations of the constrained optimization paradigm, we propose an alternative approach based upon a generative modeling framework. The key insight in our framework is that we learn a deep generative model of our target domain, and 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 $\mathbb{R}^d$.

3.1 Model overview

We seek to define an adversarial generator network, G, which specifies a conditional distribution $P(x'|x)$ over adversarial examples x' given an input datapoint x. Without loss of generality, we define a generic adversarial generator network as a combination of four components (Figure 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 $x_i \in \mathcal{X}$. For simplicity, we assume that this encoder relies on the reparameterization trick via the Gaussian distribution [29], due to its attractive sampling properties. That is, we specify 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. Following the reparameterization trick, we sample from $z \sim q_\phi(z|x)$ by drawing $\epsilon \sim \mathcal{N}(0, I)$ and returning $z = \mu_\phi(x) + \sigma_\phi(x) \circ \epsilon$, where $\circ$ denotes the elementwise product.
- A *probabilistic decoder network*, $p_\theta(\delta|z)$ that maps a sampled latent code $z \in \mathbb{R}^d$ to a perturbation δ. This function may be an arbitrary differentiable neural network.
- A *combination function* $c(\delta, x)$ that takes an input x and perturbation as input and outputs a perturbed example x'. The combination function can be as simple as adding the perturbation to the clean input—i.e., $x' = x + \delta$—or more complex such as first projecting to a candidate set.
- A *similarity function* Δ defined over $\mathcal{X}$, used to restrict the space of adversarial examples. A common choice is any l_p-norm for continuous domains or an upper bound on the number of actions performed if the domain is discrete.

The basic flow of our generator network is summarized in Figure 1: we use the encoder network to sample a latent "perturbation code" $z \in \mathbb{R}^d$ given an input $x \in \mathcal{X}$, and given this sampled code, we then generate a perturbation $\delta = p_\theta(z)$ that is combined with our original input using the combination function to generate the adversarial sample $x' = c(\delta, x)$.

This generic encoder-decoder framework for adversarial attacks can be easily adapted to different domains (e.g., text, images, or graphs) by simply specifying domain-specific variants of the four components above, with further details provided in Section 3.3. A substantial benefit of this generic encoder-decoder framework is that the attacker can make use of domain-specific prior information readily available in the input data through an appropriate choice of architecture. For example, if the input is an image, its inherent structure can be exploited by using the inductive bias of CNNs. Similarly, an appropriate choice for text data can be LSTMs and GNNs for graph-structured data.

Adversarial generalization. Unlike the constrained optimization approach, our framework also allows us to amortize the inference process: instead of performing a specific optimization for each attacked example, our trained encoder-decoder can efficiently generate a distribution of adversarial examples on arbitrary input points, even points that were unseen during training. In addition, since the output of our encoder $q_\phi(z|x)$ is stochastic, we learn a *distribution* over adversarial examples, making it possible to efficiently sample a diverse set of attacks. One key benefit of this fact is that even when one adversarial sample generated through DAGAER is unsuccessful, it is computationally inexpensive to generate another sample that may turn adversarial. In Section 4.2, we exploit this phenomenon to resample new examples in cases where our first attempt fails to attack a model, and show that this simple procedure can significantly improve attack generalization on a test set.

3.2 Training and loss function

To train our model we define a hybrid objective function that is composed of a misclassification loss, $\mathcal{L}_c$, that penalizes the model if the generated x' points are not adversarial, as well as two regularizers.

Misclassification loss. We use a max-margin misclassification loss, since it has been shown to be robust to hyperparameter settings [5]. 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. To regularize the output perturbation, we penalize the model according to the similarity $\Delta(x, x')$ between the perturbed and unperturbed points. We also add a KL-regularizer in the latent space between a uniform Gaussian prior and the distribution defined by $z \sim q_\phi(z|x)$, which prevents the latent samples z from collapsing onto one specific latent code.

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

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

Learning in this setting consists of first sampling a mini-batch of unperturbed samples and then running them through the adversarial generator G, which attempts to fool every example in the mini-batch before updating in an end-to-end fashion through stochastic gradient descent. Our approach closely resembles that of a variational auto-encoder (VAE) with a Gaussian prior on the latent space [14]. However, unlike a standard VAE, our objective function is an adversarial loss and not 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.

3.3 Implementations in different input domains

To deploy the DAGAER framework on a particular input domain, one simply needs to specify domain-specific implementations of the four key components: an encoder network, decoder network, combination function, and a similarity function.

Attacks on image classification. Following standard practice in the image domain, we define the encoder $q_\phi(z|x)$ to be a convolutional neural network (CNN) and the decoder $p_\theta(z)$ to be a deconvolutional network. The combination function is defined to be simple addition $c(\delta, x) = x + \delta$

and we define Δ to be the l_2 distance. Thus, in this setting our adversarial generator network G simply uses a generative model to output a small perturbation in continuous pixel space.

Attacks on text classification. In the text setting, we define the input $x = (x_1, ..., x_T)$ to be a sequence of T words from a fixed vocabulary $\mathcal{V}$, and we seek to generate x' as an equal-length perturbed sequence of words. Since we are focusing on attacking recurrent neural networks (RNNs), we also assume that every word $w \in \mathcal{V}$ is associated with an embedding vector $E(w) \in \mathbb{R}^d$, via a known embedding function $E : \mathcal{V} \rightarrow \mathbb{R}^d$. We use an LSTM [12] encoder $q_\phi(z|x)$ to map the sequence $(E(x_1), ..., E(x_T))$ of input word embeddings to a latent vector z. The decoder $q_\phi(z|x)$ is then defined as an LSTM that maps the latent code z to a sequence of T perturbation vectors $\delta = (\delta_1, ..., \delta_T)$, with $\delta_i \in \mathbb{R}^d$. To generate a valid sequence of words, we combine each perturbation δ_i with the corresponding input word embedding $E(x_i)$ and employ a projection function $\Pi_\mathcal{X} : \mathbb{R}^d \rightarrow \mathcal{V}$ to map each perturbed embedding to valid vocabulary word via a nearest-neighbor lookup in the embedding space.

Note, however, that the discrete nature of the projection operator $\Pi_\mathcal{X}$ represents a challenge for end-to-end training. One option is to simply define the combination function as $c(\delta_i, x_i) = \Pi_\mathcal{X}(E(x_i) + \delta_i)$ and omit the discrete projection during training (similar to [9]), but—as we will show in the experiments (Section 4.2)—this leads to sub-optimal attacks.

To address this issue, we develop a differentiable nearest neighbour attention mechanism (inspired by [28]), where for each word x_i in the input sequence x, we define the combination function $c(x_i, \delta_i)$ with a tunable temperature parameter τ as:

$$x'_i = c(\delta_i, x_i) = \Pi_\mathcal{X}\left(\sum_{w \in \mathcal{V}} \frac{\exp(\alpha_w/\tau)E(w)}{\sum_{w' \in \mathcal{V}} \exp(\alpha_{w'}/\tau)}\right) \tag{2}$$

where α_w denotes the angular distance between the perturbed embedding $E(x_i) + \delta_i$ and the embedding $E(w)$ of vocabulary word $w \in \mathcal{V}$. As with images, we use the l_2 norm on the embeddings as the similarity function.

Attacks on node classification. The final domain we consider is the setting of semi-supervised node classification using graph neural networks (GNNs) [15, 30]. We consider the challenging attack setting where the attack model can only make changes to node attributes and not the graph structure itself [38]—employing a graph convolution network (GCN) [15] 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. [38], 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 $b \in [0, 1]^N$. Again, 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 classification tasks where the input domain is one of natural images, textual data and graph-structured data. Through our experiments we seek to answer the following questions:

- **(Q1) Domain Agnostic.** Can we use the same attack strategy across multiple domains?
- **(Q2) Attack Generalization.** Is it possible to adversarially generalize to unseen examples?
- **(Q3) Diversity of Attacks.** Can we generate diverse sets of adversarial examples?

As points of comparison throughout our experiments, we consider constrained optimization-based baselines specific to the different domains we examine (and which use comparable setups). We also experiment with a simplified version of DAGAER that uses a deterministic autoencoder, rather than a variational approach (denoted DAGAER-AE). Code to reproduce our experiments is included with the submission and will be made public.

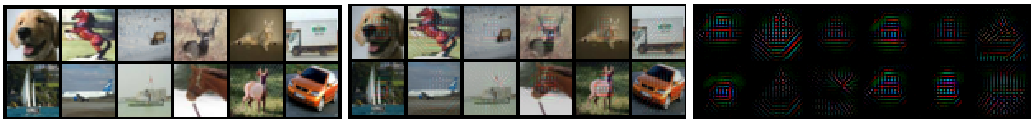

Figure 2: **Left** Unperturbed images from CIFAR10. **Middle** Attacked images with $\lambda = 0.1$. **Right**: Perturbation magnified by a factor of 10 for sake of visualization.

Table 1: Adversarial text example for the IMBD dataset. The original review was classified as negative; the adversarial example changes this classification to positive.

Original: hardly a masterpiece. not so well written. beautiful cinematography i think not. this movie wasn't too terrible but it wasn't that much better than average. the main story dealing with highly immoral teens should have focused more on the forbidden romance and why this was...

Adversarial: hardly a masterpiece not so well vanilla beautiful mystery, i think annoying this movie wasn't too terrible but it wasn't that much better than vanilla the main story anymore with highly compromising lovers, should have focused more on the forbidden romance and why this lovers

4.1 Datasets and Target Models

We attack popular neural network classification models across image, text, and graph classification using standard benchmark datasets.

Image domain. We use the CIFAR10 [16, 17] dataset for image classification (with standard train/test splits), and attack a CNN classification model based on the VGG16 architecture. The target CNN model is trained for 100 epochs with the Adam optimizer with learning rate fixed at $1e - 4$.

Text domain. We use the IMDB dataset for sentiment classification (with standard train/test splits) [21]. The target classifier is a single-layer LSTM model, initialized with pretrained GloVe embeddings [27] and trained for 10 epochs with the Adam optimizer at default settings. Note that we focus on *word-level* attacks and thus omit a detailed comparison with and discussion of character-level heuristic approaches (e.g., [18, 7]).

Graph domain. We consider two standard node classification datasets, Cora and CiteSeer [20, 4, 31]. 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 [15] 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 3.3); in the influencer setting we attack a node by changing the features of a random sample of 5 of its neighbors.[3]

4.2 Results

We now address the core experimental questions (**Q1**-**Q3**), highlighting the peformance of DAGAER across the three distinct domains.

Q1: Domain Agnostic. Our first set of experiments focus on demonstrating that DAGAER can achieve strong performance across three distinct input domains. Overall, we find that DAGAER performs competitively in the image setting (Table 2) and achieves very strong results in the text (Table 3) and graph (Table 4) domains. However, we emphasize that these results are meant to demonstrate the flexibility of DAGAER across distinct domains; an exhaustive comparison within each domain—especially for images—would necessitate a greater variety of benchmarks and evaluation settings.

For the image domain our baselines are the l_2-variants of the PGD [22] and Carlini-Wagner [5] attacks. We find that DAGAER performs competitively with the baselines with a small drop of 8% in performance for the DAGAER model and a smaller drop of 3% for the DAGAER-AE model (Table 2). Figure 5 provides an example of the perturbations generated by our approach.

For the text domain, we report the results with and without the differentiable nearest neighbor mechanism (Equation 2), with the latter also applied to the DAGAER-AE baseline for fair comparison.

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

Table 2: Attack success rate of DAGAER and DAGAER-AE on CIFAR-10 for Training and Test splits. We used $\lambda = 0.1$ and the test set results are averaged over 10 runs.

	DAGAER	DAGAER-AE	PGD [22]	Carlini-Wagner [5]
Train	92%	97%	**100%**	**100%**
Test	**81%**	67%	-	-

Table 3: Attack success rate of DAGAER and DAGAER-AE on IMDB for Train and Test splits. We used $\lambda = 0.05$ and and cap the number words changed to 15% of the sample length.

	DAGAER-AE	DAGAER-AE-Diff	DAGAER	DAGAER-Diff	FGSM [9]	DF [9]
Train	6%	44%	9%	**71%**	32%	36%
Test	18%	45%	20%	**61%**	-	-

We compare against results reported in Gong et al. [9] for their FGSM baseline and DeepFool model. We find that DAGAER outperforms both these baselines, with the differentiable nearest-neighbour mechanism being crucial to prevent overfitting (Table 3; see Table 1 for an example attack, with further details in the Appendix). Figure 4 plots the tradeoff in adversarial success between the perturbation regularization and percentage of tokens changed. As expected, more regularization induces less text change, at the expense of lower performance. Note that we do not compare against recently reported results on Alzantot et al.'s genetic attack algorithm [2], since it has not been scaled beyond a small subset of the IMDB data and requires significantly truncating the review text.

Finally, in the graph domain, we find that our DAGAER-AE baseline achieves a new state-of-the-art for *direct attacks* (when modifying node features only), outperforming the constrained optimization NetAttack approach [38] and achieving a perfect success rate on the training set (Table 4). Both DAGAER-AE and DAGAER 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. [6], since that work modifies the adjacency matrix of the graph, whereas we modify node features only.

Q2: Attack Generalization. We demonstrate the ability of DAGAER to a attack a test set of previously *unseen* instances, with results summarized under the Test columns in Tables 2-4. Since constrained optimization-based attack strategies cannot generalize to a test set of examples, we rely on our DAGAER-AE approach as a strong baseline. In all three domains DAGAER is able to generalize effectively, outperforming DAGAER-AE in the image setting and in the text setting. For graph attacks we observe that DAGAER has marginally better generalization ability for influencer attacks while DAGAER-AE performs better by a similar margin in the easier direct attack setting.

Q3: Diversity of Attacks. DAGAER exhibits comparable (or marginally worse) performance compared to our deterministic autoencoder (DAGAER-AE) approach in terms of raw success rates. However, a key benefit of the full DAGAER 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 3, DAGAER can produce adversarial samples for clean inputs that were originally classified correctly, significantly boosting generalization performance. Specifically, we observed that 71% of correctly classified test samples could be successfully attacked for CIFAR10 after 100 resamples. Similarly, for IMDB there is an average absolute improvement of 60%. For the graph setting 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 Further Related Work

Here, we provide a brief discussion of related work that we have not previously discussed (e.g, in Sections 1 and 4), and that is highly relevant to our proposed framework. For more comprehensive overviews of recent advancements in adversarial attacks, we direct the interested reader to survey papers for adversarial attacks in the image [1], text [35], and graph domains [32].

Table 4: Attack success rate of DAGAER and DAGAER-AE on Graph Data. For Influencer attacks we only consider the training set. We used $\lambda = 0.01$ and results are averaged over 5 runs.

	Cora Train	Cora Test	CiteSeer Train	CiteSeer Test
DAGAER-Direct	88 %	91 %	81%	82%
DAGAER-AE-Direct	**100%**	**93%**	**100%**	**92%**
Zugner-Direct	99%	-	99%	-
DAGAER-Influencer	**62%**	-	**54 %**	-
DAGAER-AE-Influencer [38]	60%	-	52%	-
Zugner-Influencer [38]	33 %	-	38%	-

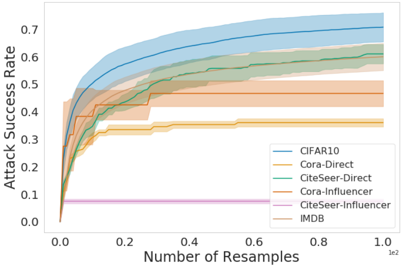

Figure 3: Attack Success Rate when resampling only failed adversarial examples in the test set.

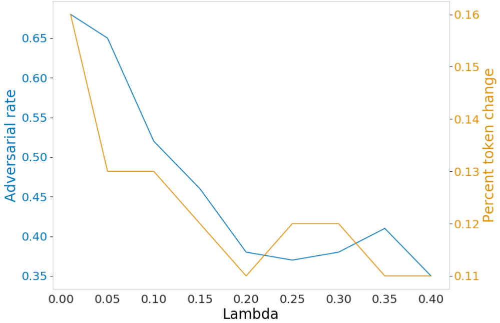

Figure 4: Tradeoff in attack success rate and word error rate with respect to λ.

Adversarial attacks using parametric models. Similar to our approach, there is relevant previous work on generating adversarial images using parametric models, such as the Adversarial Transformation Networks (ATN) framework and its variants [3, 36]. However, unlike our approach, they are specific to the image domain and do not define a generative distribution over adversarial examples.

Adversarial attacks in latent space. There is a burgeoning literature of attacks that are crafted in a latent space, similar to the framework proposed here. For instance, Zhao et al. [37] propose a framework to generate "natural" adversarial examples that lie on the data manifold. However, unlike our work, their "natural adversary" requires a search procedure to generate an adversarial example and does not facilitate an efficient generation of diverse attacks like DAGAER. Concurrent to our work, Li et al. [19] introduced $\mathcal{N}$Attack to produce distributions of adversarial examples for blackbox image-based attacks. While similar in spirit to our framework and relatively efficient, $\mathcal{N}$Attack is specific to the image domain and operates within a constrained optimization paradigm—requiring additional rounds of optimization to generate adversarial examples on unseen data.

6 Discussion and Conclusion

We present DAGAER, a unified framework for constructing domain agnostic adversarial attacks. We deploy our strategy on three domains—images, text and graphs—and 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 DAGAER makes our framework 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.

In terms of limitations and directions for future work, we considered only a simple latent code which is a reparametrized Gaussian. Exploring and extending DAGAER to more complex distributions using modern generative modelling methods, such as normalizing flows, is a natural direction for future work. Extending DAGAER to the blackbox setting (e.g., using relaxed gradient estimators) is another fruitful direction for future work. Finally, while this work demonstrates the significant promise of a generative approach to adversarial attacks, further empirical studies, e.g., using various adversarial defense strategies, are necessary to more completely characterize the pros and cons of this approach compared to the constrained optimization paradigm.

Acknowledgements

The authors would like to thank Maxime Wabartha, Ariella Smofsky, 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. 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. *arXiv preprint arXiv:1801.00553*, 2018.

[2] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018.

[3] Shumeet Baluja and Ian Fischer. Adversarial transformation networks: Learning to generate adversarial examples. *arXiv preprint arXiv:1703.09387*, 2017.

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

[5] 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.

[6] 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.

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

[8] 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.

[9] Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*, 2018.

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

[11] 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.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

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

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

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[17] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

[18] 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.

[19] Yandong Li, Lijun Li, Liqiang Wang, Tong Zhang, and Boqing Gong. Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. *arXiv preprint arXiv:1905.00441*, 2019.

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

[21] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[28] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In *Advances in Neural Information Processing Systems 31*, pages 1087–1098, 2018.

[29] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *31st International Conference on Machine Learning*, 2014.

[30] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.

[38] 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

We summarize the application of DAGAER framework to the different domains considered in the paper. 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.

Table 5: Summary of the different components to applying our framework to different input domains

ATTACK DOMAIN	INPUT TYPE	ENCODER	DECODER	δ	Δ	$c(x, \delta)$
IMAGE	PIXELS	CNN	CNN	PIXELS	l_2	$x + \delta$
TEXT-EMB	WORD EMB	LSTM	LSTM	WORD EMB	l_2	$x + \delta$
TEXT-TOKEN	DISCRETE TOKEN	LSTM	LSTM	WORD EMB	l_2	$\Pi_{\mathcal{X}}(x + \delta)$
GRAPH-DIRECT	NODE EMB	GCN	MLP	NODE EMB	l_2	$x + \delta$
GRAPH-INFLUENCER	NODE EMB	GCN	MLP	NODE EMB	l_2	$x + b \cdot \delta$

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

Image.

Table 6: Hyperparameter and experimental details for the image domain

Parameter	Description	Value
Generator Architecture	DenseNet121	-
Latent size	AE and DAGAER latent posterior size	50
Noise radius	Standard deviation for Gaussian noise	0.2
Learning rate	SGD Learning rate for the target LSTM model	0.01
Optimizer	DAGAER optimizer	Adam [13]
λ	L2 regularization coefficient	0.1
Batch size	For training and testing	256
GPU	For training and testing	2 Nvidia 1080 Ti
Similarity function	Metric used for Δ	l_2-distance

Text.

For DAGAER-Diff we began training with a softmax temperature τ 1 for the differentiable nearest neighbour mechanism. After every 20 batches, we decreased τ with a decay rate of 0.9 until τ reached a minimum 0.01.

Table 7: Hyperparameter and experimental details for the text domain

Parameter	Description	Value
LSTM size	Hidden layer size for target model	300
Gradient clipping norm	The clipping norm for the target LSTM model when training	1
Latent size	AE and DAGAER latent posterior size	50
Noise radius	Standard deviation for Gaussian noise	0.2
Learning rate	SGD Learning rate for the target LSTM model	0.01
Optimizer	DAGAER optimizer	Adam [13]
λ	L2 regularization coefficient for embedding perturbation	0.05
Batch size	For training and testing	128
Embedding size	Glove embeddings, determined by target model	300
GPU		2 Tesla P100
Max input length	Maximum length for IMDB text	200
Similarity function	Metric used for Δ	l_2-distance
Distance function	Metric used for the nearest neighbour function	Angular distance

Graph.

Table 8: Hyperparameter and experimental details for the Graph domain

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 DAGAERtraining	200
Latent size	AE and DAGAER latent posterior size	50
Dropout Ratio	Used for Regularization	0.5
$\|\mathcal{T}\|$	Number of Target Nodes	40
Learning rate	Learning rate for G	0.01
Optimizer	DAGAER optimizer	Adam [13]
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

B Additional Results

Training resutls on IMDB.

Table 9: Attack success rate of DAGAER and DAGAER-AE on IMDB for Train and Test splits. We used $\lambda = 0.05$ and the test set results and cap the number words changed to 15% of the sample length. “Emb” rows correspond to success rate where the target model receives perturbed embeddings as input (no tokens), whereas “Proj” rows correspond to the success rate where the perturbed embeddings are first projected to nearest neighbour embeddings, and then passed to the target model. The latter setup correspond to a realistic application where only adversarial tokens are used as input. Additionally, we used 20% of the training set for validation.

	DAGAER-AE	DAGAER-AEDiff	DAGAER	DAGAER-Diff	FGSM [9]	DF [9]
Train-Emb	**96**	86%	94%	58%	88%	45%
Train-Proj	6%	44%	9%	**71%**	32%	36%
Test-Emb	72%	**76%**	70%	56%	-	-
Test-Proj	18%	45%	20%	**61%**	-	-

Adversarial Image Examples.

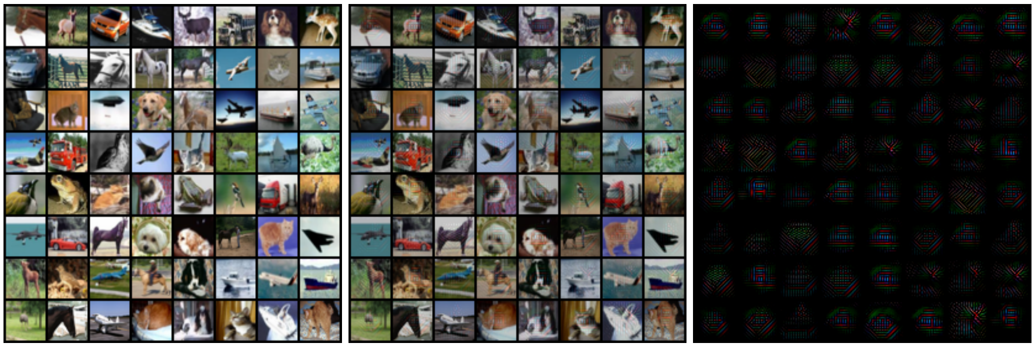

Figure 5: **Left** Unperturbed images from CIFAR10. **Middle** Attacked images with $\lambda = 0.1$. **Right**: Perturbation magnified by a factor of 10 for sake of visualization.

Adversarial Text Examples.

Table 10: Adversarial text example for the IMBD dataset. The original review was classified as negative; the adversarial example changes this classification to positive.

Original: i saw this film yesterday. i must admit, it weren’t my cup of tea. although it’s supposed to be a horror movie of its kind. but as i was watching this, i was ’this movie isn’t making any sense at where on earth did this guy in the dark coat came from? where were the two guys were going when they left the girls where on earth did a shark came out /><br />all these elements in this film somehow didn’t add up.

Adversarial: i saw this film arid i must listings it is, my cup of (overall although it’s supposed to be a horror movie of its 7 but as i was watching this, i was chief movie isn’t making any sense at where on earth did this guy in the dark arid came redspin-offs where were the two guys were going when they left the girls where on earth did a 7 came out /><br andre these elements in this film somehow didn’t add up.