UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2024-047
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 28/06/2024 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG EN INFORMATIQUE

by

Salijona DYRMISHI
Born on 20th December 1996 in Tërpan, Berat (Albania)

ENHANCING MACHINE LEARNING SECURITY: THE SIGNIFICANCE OF REALISTIC ADVERSARIAL EXAMPLES

Abstract

Adversarial attacks pose a significant security threat in Machine Learning (ML), employing subtle, invisible perturbations on original examples to craft instances that deceive model decisions. While extensively studied in computer vision and diverse domains such as credit scoring, cybersecurity, cyber-physical systems, and natural language processing, recent findings reveal limitations in traditional adversarial attacks. These approaches often yield examples that lack realism, failing to map to real-world objects or adhere to imperceptibility requirements. The field of realistic adversarial attacks and their implications on the robustness of real-world systems is currently under-explored in the literature. Through this thesis we demonstrate the importance of realism in adversarial attacks, the conditions on which these attacks are realistic, and propose new strategies to upgrade current attacks from unrealistic to realistic.

As a first contribution, we shed light on the importance of producing realistic adversarial examples when hardening models against realistic attacks. We use three real-world use cases (text classification, botnet detection, malware detection) and seven datasets in order to evaluate whether unrealistic adversarial examples can be used to protect models against realistic examples. Our results reveal discrepancies across the use cases, where unrealistic examples can either be as effective as the realistic ones or may offer only limited improvement. Second, to explain these results, we analyze the latent representation of the adversarial examples generated with realistic and unrealistic attacks. We investigate the patterns that discriminate which unrealistic examples can be used for effective hardening.

As a second contribution, we evaluate the realism of the adversarial examples generated by textual attacks. The current attacks ignore the property of imperceptibility or study it under limited settings. This entails that adversarial perturbations would not pass any human quality gate and do not represent real threats to human-checked NLP systems. To bypass this limitation and enable proper assessment (and later, improvement) of NLP model robustness, we have surveyed 378 human participants about the perceptibility of text adversarial examples produced by state-of-the-art methods. Our results underline that existing text attacks are impractical in real-world scenarios where humans are involved.

This contrasts with previous smaller-scale human studies, which reported overly optimistic conclusions regarding attacks' success. Through our work, we hope to position human perceptibility as a first-class success criterion for text attacks, and provide guidance for research to build effective attack algorithms and, in turn, design appropriate defence mechanisms.

As a final contribution of this thesis, we enhance adversarial attacks based on Deep Generative Models (DGMs) by introducing a constrained layer to generate more realistic examples for tabular datasets. DGMs are widely utilized for synthesizing data that mirrors the distribution of original data, serving purposes like dataset augmentation, fairness promotion, and sensitive information protection. DGMs have also been leveraged for adversarial generation processes (AdvDGMs). Despite their proficiency in modeling complex distributions, DGMs often produce outputs that violate background knowledge expressed through numerical constraints, resulting in lower success rates for AdvDGMs. To address this limitation, we transform DGMs into Constrained Deep Generative Models (C-DGMs) using a Constraint Layer (CL) that repairs violated constraints. Moreover, we extend these C-DGMs to C-AdvDMs for generating realistic adversarial examples. Our experiments demonstrate that integrating this constraint layer, which incorporates background knowledge, not only enhances the quality of sampled data for machine learning model training but also improves the success rate of AdvDGMs. Notably, these enhancements are achieved without compromising sample generation speed.

Acknowledgments

This thesis was finalized with the support of many, but I owe my gratitude first and foremost to my supervisor, Dr.Maxime Cordy. With his guidance, I got inspiration for new ideas and improved many of my research skills, including project conception, critical thinking, and scientific writing. I can testify that he could write great scientific texts before the large language models appeared. But I am most grateful for the independence he gave me over time and the trust to pursue activities beyond pure research. Thank you Maxime, I consider myself lucky!

I am deeply grateful to my co-authors (to be Dr.)Thibault Simonetto, Dr.Salah Ghamizi, (to be Dr.)Mihaela Stoian and Dr.Eleonora Giunchiglia who have played different roles —sometimes as mentors, peers, reviewers, or fellow PhD journey companions. I have learned great things from them, and there is no doubt that their contributions have helped shape the work you are about to read. I do not dare to forget here past and present colleagues from SerVal (SnT) with whom I had great discussions about academia and learned about diverse topics far from this thesis. My appreciation goes to the group leadership who facilitated this good work environment, for their open mind and their efforts to facilitate travel and collaborations. It is now time to thank the major financial contributor to this thesis through an AFR Grant, the Luxembourg National Research Fund (FNR).

On a personal note, many people not necessarily linked to the research world have contributed indirectly to the completion of this thesis, including friends and family. They deserve my gratitude for understanding the busy periods and sticking around no matter the long distances between us. I know they are very proud of this achievement at all times. My family instilled in me the values of hard work, honesty, and modesty, which guided me to humbly seek knowledge through this thesis in often unknown directions. So I say *Faleminderit!* to my parents *Eqerem* and *Alime*, my brother *Ledio*, and my grandmas *Barie* and *Minushe*.

Finally, this thesis brought occasionally dark times —hopelessness over failed experiments and unexplainable results, a sense of purpose loss, or many late nights of work. *Faleminderit!* immensely to my husband, *Sri*, who saw and lifted me through these difficult periods. You were right, I would enjoy a PhD journey, but I am not sure if I would reach the finish line without your support.

Contents

1 Introduction

This chapter sets the stage by discussing the context and challenges of this dissertation. First, we introduce adversarial attacks, a key security issue in machine learning. Then, we examine the realism of current adversarial testing techniques. Second, we highlight the challenges and provide open directions on operating in realistic settings for adversarial testing. Finally, we outline the contributions of this dissertation.

Contents

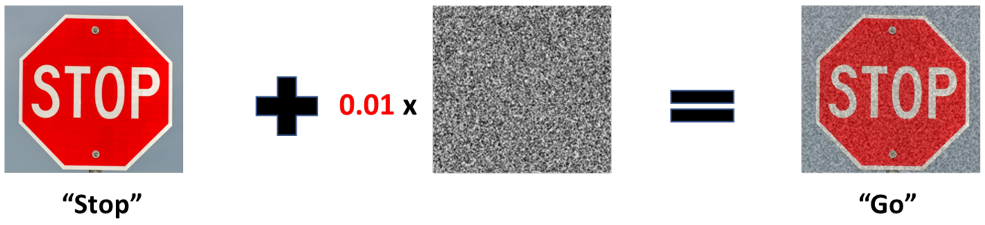
1.1 Context

Machine learning (ML) models find a vast application in today's world. Their ability to analyze large amounts of data, extract meaningful insights, and make instant decisions has been a key factor for several industries (i.e finance, cybersecurity, medical, transportation) to automate their tasks, optimize operations, and enhance decision-making. The rapid growth of ML can be attributed to several factors, including the increase in available data, advancements in computing power, and the development of sophisticated algorithms. Moreover, the versatility of ML enables it to tackle a wide range of problems, from image recognition to natural language processing, making it applicable in diverse scenarios. The recent advancements in generative AI, coupled with increased accessibility, have also contributed to this growing popularity. User-friendly interfaces based on natural language are further breaking down barriers between users and AI. This accessibility has facilitated the widespread adoption of ML technologies, as demonstrated by platforms like ChatGPT, which gained over 1 million users in just five days.

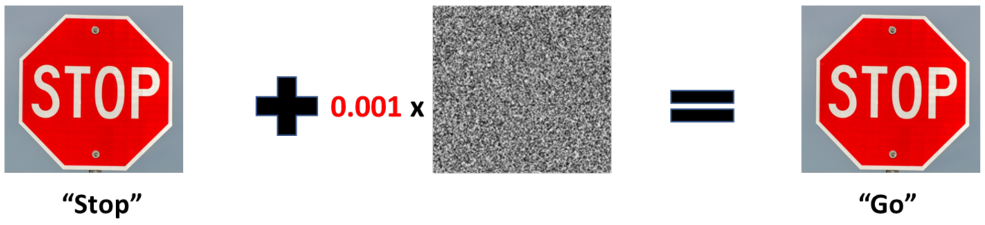
Throughout the ML lifecycle, these learning systems are faced with a variety of security threats. These threats occur from the initial stages of data acquisition and preprocessing, where adversaries may attempt to extract sensitive information or inject malicious data to compromise model integrity, to the deployment phase, where models are susceptible to attacks such as evasion and model stealing.

Several frameworks aim to model the risks associated with AI and machine learning systems. One of them, the MITRE ATT&CK® framework, extends to cover AI-specific threats, providing a structured taxonomy of adversary behaviours that target machine learning deployments. These threats pose significant risks to the integrity and reliability of machine learning systems. Occasionally they surface in the news, causing fear and doubts about the trustworthiness of ML to end users. Notably, the EU AI Act, the first regulation on Artificial Intelligence, acknowledges these risks, underlining the necessity of addressing them comprehensively.

Adversarial attacks are considered one of the most critical security threats in ML [XYW+20; LLZ+18]. These attacks that occur at inference time produce *adversarial examples*, which are minimally altered original examples that do not change the semantics for humans but cause the models to output wrong decisions. These alterations, although appearing minimal, can drastically change model's decision leading to potentially severe consequences for various domains. For instance, in image recognition tasks, imperceptible modifications to images can cause models to misclassify objects like traffic signs leading to safety hazards in autonomous vehicles (see Figure 1.1). Similarly, in natural language processing, adversarial attacks can manipulate text aiming to bypass content moderation filters such as spam, hate speech and fake news detectors. For example, consider the sentence "*People of X ethnicity are all <u>criminals</u>.*" initially classified as hate speech by an



(a) Adversarial example with **high levels of perturbation**



(b) Adversarial example with **low levels of perturbation**

Figure 1.1: An illustration of an adversarial example for an image classification task. Carefully crafted perturbations change the decision of ML model from "Stop" to "Go". Low levels of perturbation in b) makes the adversarial modifications imperceptible by humans.

ML system. An adversary’s goal would be to minimally alter the sentence while preserving its meaning, resulting in something like *"People of X ethnicity are all delinquents."* which the system identifies as non-hate speech.

To enable the secure deployment of ML models in the real world, it is essential to properly assess their robustness to adversarial attacks, a process often known as adversarial testing. A common way to assess robustness is to use adversarial generation techniques to craft adversarial examples from a set of original ones and compute the model accuracy on these adversarial examples. Adversarial testing techniques operate in two distinct spaces: problem space and feature space. In the problem space, modifications are made directly to real-world objects, such as altering code in software applications. Meanwhile, the feature space involves manipulating the numerical representations of these real-world objects that are fed into machine learning models. Many existing attacks primarily target the feature space due to its computational efficiency and/or the complexity of altering real-world objects in problem space. However, depending on the presence of an inverse mapping between the feature space and the problem space, these techniques can ultimately generate a problem space adversarial example from the one created in the feature space.

Recent studies [GCG^{+}20; TWT^{+}20], have revealed a significant limitation in many current adversarial testing techniques: the generated adversarial examples often lack realism, meaning they do not correspond to realistic real-world objects. As a result, while these attacks may successfully deceive machine learning models in controlled environments, their efficacy in assessing the robustness of models in practical settings is compromised. This problem arises from attacks perturbing input samples without considering the diverse range of domain constraints that govern real-world data. In the problem space, domain constraints manifest as a set of permissible transformations that maintain the functionality and integrity of real-world objects. For instance, in software development, these constraints might dictate the acceptable sequence of API calls or permissible modifications to source code files. These constraints are essential for ensuring that the resulting artifacts remain valid and functional within their respective domains. On the other hand, in the feature space, constraints may arise either as a direct result of mapping from the problem space or as unintended side-effects of feature engineering processes. For example, in a credit scoring system, a constraint could stipulate that the average number of transactions per month cannot exceed the total number of transactions, reflecting the inherent characteristics and limitations of the data being used for model training. Adversarial attacks are not aware of the existence of such domain constraints by defaults, therefore many of them fail to generate realistic examples. This highlights the importance of considering domain-specific constraints when designing adversarial attack techniques and evaluating model robustness.

The vulnerabilities discovered through adversarial testing are addressed through the adversarial hardening process, which brings robustness improvements against adversarial attacks. This approach involves implementing a range of techniques (adversarial training, regularization, and model ensembling) aimed at fortifying models' resilience to adversarial perturbations in input data. The variations of adversarial training techniques, in particular, involve augmenting the training data with adversarial examples, forcing the model to learn to resist perturbations during the learning process. Integrating these adversarial hardening techniques into the development of ML models, can strengthen their security and mitigate the impact of adversarial threats in real-world applications.

1.2 Challenges

We have identified three main challenges in assessing and improving the robustness of ML systems in a real-world setting:

1. **High cost of defense against realistic adversarial attacks**
   Defending against adversarial examples typically involves integrating them into the training process with their correct labels, enabling the model to adjust its learning process accordingly. However, generating realistic adversarial examples proves to be expensive across many domains. This occurs either when there is no direct mapping between the feature and the problem space or

when addressing domain constraints within the feature space before mapping back to the problem space requires significant overhead. As a result, only a limited number of realistic adversarial examples can be generated within a reasonable time. If we couple this process with the inherent training costs, the costs of hardening models can potentially become unaffordable.

2. **Lack of universal understanding on the realism of adversarial examples**
   Adversarial attacks aim to cause incorrect decisions, maintain functionality, and avoid detection. However, assessing the realism of examples generated by such attacks presents significant challenges, as verifying their real-world feasibility is often not possible. One approach to ensure realism in the problem space relies on object transformations that maintain functionality but these may not cover all possibilities, narrowing the attack search space. Another faster method is checking if examples adhere to domain constraints in feature space but this method can fail due to lack of or improper constraint definitions. The challenge becomes more complex when human interaction with the ML system is involved. Imperceptibility, which is closely tied to human perception, remains an open question in terms of realism. There is ongoing debate about the applicability of L_p norms, which control the magnitude of perturbation, beyond the image domain to other areas such as spam filtering, medical recommendations, and credit approval. While imperceptibility in images can be easily validated by applying perturbations within small L_p norms, as shown in Fig 1.1, it is underexplored and often overlooked in other domains, especially in natural language processing (NLP). Adversarial attacks in NLP frequently achieve high success rates against the latest models but often neglect or only partially explore imperceptibility. This suggests that adversarial perturbations may not pass human quality checks and do not pose real threats to NLP systems gated by humans.

3. **Lack of general methods to generate more realistic adversarial examples in feature space**

   Running attacks in problem space is a complex task, especially since we need to generate a large number of adversarially manipulated objects to test our models. This process involves defining feasible transformations while ensuring that they maintain feasibility. Engineering efforts, such as operating malware sandboxes, and design considerations, including domain-specific adaptations for attacks, further complicate the task, often resulting in increased runtime. In contrast, manipulating feature vectors offers a quicker and simpler alternative. However, while some approaches integrate constraints like feature types and non-modifiable features to enhance realism, these constraints often fail to account for the intricate relationships between features themselves. Additionally, in parallel to this thesis, some more generic methods have been developed to include constraints in the loss function. Nevertheless, these methods rely on the optimization function and cannot guarantee constraint compliance all the time.

1.3 Overview of contributions

This dissertation addresses the above challenges and brings three contributions:

- **An empirical study on the impact of unrealistic hardening against realistic adversarial examples (Chapter 4).** To address the first challenge, we use three real-world use cases (text classification, botnet detection, malware detection) and seven datasets to evaluate whether unrealistic adversarial examples can be used to protect models against realistic examples at a reduced cost. Our results reveal discrepancies across the use cases, where unrealistic examples can either be as effective as the realistic ones or may offer only limited improvement. To explain these results, we analyze the latent representation of the adversarial examples generated with realistic and unrealistic attacks. We shed light on the patterns that discriminate which unrealistic examples can be used for effective hardening.
- **An extensive human study to evaluate the realism of adversarial text as perceived by humans (Chapter 5).** To address the second challenge, we have systematized the evaluation criteria for the imperceptibility of adversarial text and asked 378 human participants to evaluate adversarial examples produced by state-of-the-art methods based on these criteria. Our results underline that existing text attacks are impractical in real-world scenarios where humans are involved. Through this contribution, we hope to position human perceptibility as a first-class success criterion for text attacks, and provide guidance for research to build effective attack algorithms and, in turn, design appropriate defence mechanisms.
- **A novel method to generate more realistic adversarial examples in feature space(Chapter 6).** To address the third challenge, we adapt adversarial attacks based on Deep Generative Models (DGMs) to ensure they generate adversarial examples satisfying background knowledge expressed as numerical constraints. Adversarial DGMs have the potential to reduce adversarial training costs since once trained, the adversarial generation process goes only in a single forward pass contrary to other iterative attacks. However, current DGMs are effective for generating adversarial examples by capturing complex distributions, but they do not guarantee the satisfaction of domain constraints, reducing their effectiveness as adversarial techniques. We address this by transforming DGMs into Constrained Deep Generative Models (C-

DGMs), integrating a Constraint Layer (CL)[1] that ensures generated samples meet given constraints. Moreover, we extend these C-DGMs to C-AdvDMs for generating realistic adversarial examples. Our experiments demonstrate that integrating this constraint layer, which incorporates background knowledge, not only enhances the quality of sampled data for machine learning model training but also improves the success rate of AdvDGMs. Notably, these enhancements are achieved without compromising sample generation speed.

[1]The CL layer was designed and implemented by Mihaela Stoian in the context of our broader collaborative work. For a detailed description of the context, please refer to Appendix 8.3.1

2 Background

This chapter introduces the technical concepts of adversarial attacks and hardening methods, which are relevant throughout this thesis.

Contents

We consider a multi-class classifier $H : \mathcal{X} \longrightarrow \mathcal{Y}$ that maps an m-feature vector $x = \{x_1, x_2..., x_m\} \in \mathcal{X}$ to a class $y \in Y$. The function $h : \mathcal{X} \rightarrow \mathbb{R}$ predicts a probability distribution for Y. The predicted class $H(x)$ can be derived as the class with the highest probability, such that $H(x) = \underset{i \leq m}{argmax} \quad h_i(x)$.

2.1 Adversarial Attacks

An adversary aims to change the predicted class $H(x)$ by the classifier. In the case of targeted attacks, the aim is to flip the prediction to a desired class that is predefined. In the case of untargeted attacks, it is sufficient that the label is different from its original prediction. For simplicity, we formalize below the case of targeted attacks only. Another distinction concerns the dimension the adversary works in. We can classify all attacks into feature-space attacks (that alter input features) or problem space attacks (that directly manipulate domain objects). This refers to the case where the feature mapping between feature space and problem space is not differentiable and not invertible.

2.1.1 Feature space attacks

The attacker in feature space modifies directly the feature vector x_0 that is fed to the machine learning classifier $h(x)$. We distinguish between (traditional) unconstrained attacks and (domain-specific) constrained attacks.

Unconstrained feature-space attacks. Given an initial feature vector x_0, the attacker aims to maximize the probability that the adversarial feature vector $\hat{x_0} = x_0 + \delta$ is classified as a predefined target class $t \neq H(x_0)$, where δ is a perturbation vector. When performing an unconstrained attack, the adversary can perturb every feature as long as the generated adversarial example remains sufficiently close to the original example. The most popular metrics to measure similarity are L_p norms that come from computer vision, however, in other domains specific metrics can also be defined (e.g. Cartella et al. [CAF$^+$21] define a metric for credit fraud detection that considers feature importance and human checks). We, then, define an unconstrained adversarial attack objective as:

$$\begin{aligned} \text{minimize} \quad & D(x_0, \hat{x_0}) \\ \text{such that} \quad & H(\hat{x_0}) = t \end{aligned} \tag{2.1}$$

where D is the distance metric of choice.

Constrained feature-space attacks. Many classification datasets are subject to constraints that come from inherent domain properties or from the way features are engineered. Different forms of constraint exist, including feature immutability (the adversary cannot alter specific features), data types, linear/non-linear relationships between multiple features, etc. These constraints pose additional restrictions for the adversarial examples to be considered valid (in addition to the perturbation size). Hence, constrained attacks perturb the original example in a direction that is valid with respect to the constraints. We denote the set of constraints by Π and the set of feasible examples by $X_\Pi = \{x \in X | x \vDash \Pi \in \Pi\}$. When Π exactly translates all the constraints that exist in a domain, all examples in X_Π may be produced in reality. Constrained adversarial attacks generate only examples in X_Π. Hence, the attack objective takes the form:

$$\begin{aligned} \text{minimize} \quad & D(x_0, \hat{x_0}) \\ \text{such that} \quad & H(\hat{x_0}) = t \\ & \hat{x_0} \in X_\Pi \end{aligned} \tag{2.2}$$

2.1.2 Problem space attacks

In this setting instead of dealing with numerical feature vectors, problem-space attacks directly manipulate physical (e.g. printed patches, sensors, sounds) or digital (e.g. pdf files, Android apps) domain objects. Let Z be the set of domain objects. The feature mapping function $\phi : Z \longrightarrow X$ maps any $z \in Z$ to a feature representation $x \in X$. Problem space attacks aim to find a valid sequence of domain-object transformations $T = T_n \circ T_{n-1} \circ ... \circ T_1$, such that $T(z)$ satisfies some problem space constraints γ. These constraints γ encapsulate both general adversary concerns (e.g. similarity or preserved semantics) and domain-specific considerations (e.g. plausibility) [PPC+20]. The problem-space attack objective is then:

$$\begin{aligned} \text{minimize} \quad & |T| \text{ where } T \in \mathcal{T} \\ \text{such that:} \quad & H(\phi(T(z))) = t \\ & T(z) \vDash \gamma \end{aligned} \tag{2.3}$$

where $\mathcal{T}$ is the space of sequences of available transformations and $|T|$ is the length of a sequence T.

2.2 Hardening methods

There are different approaches to harden a model against adversarial attacks. We focus on the methods that involve the inclusion of adversarial examples in the training process of the classifier $H(x)$ based on the original training dataset $D = \{(x_i, y_i)_{i=1}^N\}$. We name such methods *adversarial hardening* and distinguish between three categories.

2.2.1 Adversarial training

[SZS+13] is the most standard way to harden models. In application, it generates adversarial examples using an efficient method (e.g. FGSM [GSS14]) and include these examples during the training process of the model, mixing the adversarial examples with the original data over the training epochs. The most established strategy [MMS+17] is to generate each time the worst-case adversarial examples (that achieve the highest loss) and, then, update the model learnable parameters to minimize the classification error over these adversarial examples. That is, adversarial training solves the following min-max optimization problem:

$$min_{\theta} \quad \mathcal{E}(x,y) \sim D\left[max_{\delta \in S} L(\theta, x+\delta, y)\right] \tag{2.4}$$

where $(x, y) \sim D$ represents training data sampled from the distribution D, $\delta \in S$ is the maximum allowed perturbation, L is the model loss and Θ is the model hyperparameters.

2.2.2 Adversarial retraining

[CWC20] is among the simplest techniques to defend against adversarial attacks. We use an attack A over the training dataset D to produce an adversarial training set $D^a = \{(\hat{x_i}, \hat{y_i})_{i=1}^{N}\}$. We join the original and the adversarial training sets and retrain the classifier $H(x)$ over this augmented set $D^* = D \cup D^a$. It is to be noted that this is different from adversarial training as proposed by [SZS+13], where new adversarial examples are generated and mixed with the original training set *continuously.* Adversarial retraining is suited for both traditional machine learning algorithms and neural networks.

2.2.3 Adversarial fine-tuning

[JSW20] is a computationally efficient alternative to adversarial training. Indeed, while adversarial training resists well even newly emerging attacks [ACW18], it is also very costly. On average, it needs 8–10 times more computational resources than the normal training of a neural network [JSW20]. Therefore, some authors propose to perform adversarial fine-tuning, which is a mix of standard training and adversarial training. The idea of the adversarial fine-tuning is that we train for $e = e' + e''$ epochs: e' epochs of standard training (using original training data) and, then, e'' epochs of adversarial training (using only adversarial examples).

3

Related Work

This chapter discusses the existing work related to adversarial attack techniques and their considerations on generating realistic examples.

Contents

3.1 Feature space attacks

Most of the constrained attacks in feature space are an upgrade of traditional computer vision attacks like JSMA [PMJ$^+$16], C&W [CW17] or FGSM [GSS14]. The majority of these attacks are evaluated in Network Intrusion Detection Systems (NIDS) [CO19; SPW$^+$20; TPS20; TWT$^+$20; SHP$^+$21] and few other domains: Cyber-Physical Systems [LLY$^+$20], Twitter bot detection and website fingerprinting [KHS$^+$18], and credit scoring [GCG$^+$20; LMK$^+$20; CAF$^+$21]. Every attack handles different types of constraints, resulting in loosely realistic to totally realistic adversarial examples.

Kulynych et al. [KHS$^+$18] proposes a graphical framework where the nodes represent the feasible examples and the edges represent valid transformations for each feature. Despite the results, this A* based attack is practically infeasible in many large datasets due to its time complexity. Levy et al. [LMK$^+$20] updates JSMA attack to take into account immutable features and feature distribution. Cartella et al. [CAF$^+$21] proposes some modifications to three existing attacks: Zoo, Boundary attack, HopSkipJump. The modifications consider the following factors: feature modifiability, feature data types, and feature importance. All these three attacks [KHS$^+$18; LMK$^+$20; CAF$^+$21] mentioned above consider only constraints on individual features and neglect inter-feature relationships, therefore there is a high risk that attacks generate totally unrealistic examples. Another attack which considers only individual features was proposed by Teuffenbach et al. [TPS20]. They divided the features in groups based on the difficulty to attack them. They extend C&W attack to consider weights in these groups and favor modifications on low weighted features, i.e. the independent features. This approach does not guarantee that the generated adversarial examples respect constraints. Instead, it hopes that only modifications in independent features will be enough to cause a wrong prediction.

Among the attacks that consider feature relationships are the ones introduced in [TWT$^+$20], [LLY$^+$20], [SPW$^+$20], [SHP$^+$21], [CO19], [GCG$^+$20], [SDG$^+$21a]. Tian et al. [TWT$^+$20] update IFGSM so that the size and direction of the perturbation are restricted to respect feature correlations. Despite this method being among the few that use a large variety of constraints types, it can not handle relationships of more than two features. As a result, their constrained IFGSM attack generates realistic examples only in domains where there are no relationships between more than two features. Li et al. [LLY$^+$20] identify a series of linear constraints related to sensor measurements for two Cyber-Physical Systems domains and craft a best-effort search attack that would respect these constraints. Their approach handles non-linear constraints and a limited type of simple non-linear relationships.

Sheatsley et al. [SPW$^+$20] integrate constraint resolution into JSMA attack to limit feature values in the range dictated by their primary feature. Later on, for the same case study, Sheatsley et al. [SHP+21], learned boolean constraints with Valiant algorithm and introduced Constrained Saliency Projection (CSP), an improved iterative version of their previous attack. The bottleneck of this attack is the process of learning the constraints. Its runtime scales exponentially with the number of features. Chernikova et al. [CO19] designed a general attack framework named FENCE for constrained application domains. The framework includes an iterative gradient attack that updates features based on their family dependencies. FENCE is one of the attacks that handles the largest type of constraints, resulting in realistic constrained feature space examples. Ghamizi et al. [GCG+20] propose a genetic algorithm that encodes constraints as objectives to be optimized. They support a variety of linear, non-linear, and statistical constraints as long as they can be expressed as a minimization problem. This ensures that the generated constrained examples are realistic. Later, Simonetto et al. [SDG+21a] followed a similar approach as [GCG+20] to propose a generic framework for generating constrained adversarial examples.

You can find a complete list of constrained adversarial attacks and their properties in Table 3.1.

3.2 Problem space attacks

Problem space attacks design and application are complex. Generally, it is easier to deal with digital objects than tangible ones, therefore there are more problem space attacks in the digital world. [AKF+18] uses reinforcement learning to modify binary Windows PE files to evade Windows malware detection engines. The generated adversarial files are not checked if they are valid, therefore some files can be corrupt. [CSD19] uses the same set of operations as [AKF+18] combined with Genetic Algorithm to generate adversarial Windows PE files. In the contrary to [AKF+18], Castro et al. evaluate in each iteration if the modified samples are still functional in order to avoid corrupt files. The generated files are valid and preserve their maliciousness. [PPC+20] introduces a new problem space attack for android malware detection. They initially collect a set of bytecodes (gadgets) from benign files through program slicing. At the attack stage, they select the most important features for the classifier and gadgets that include those features. The gadgets are then implanted on the malicious file. By the design described above, the generated files are functional. [SEZ+20] introduces Imperio, an attack that produces adversarial audio examples for speech recognition systems. The particularity of the Imperio is that the audio can be played in different room environments without the need to be tuned for that specific environment. Additionally, the attack does not require a direct line-of-sight between speaker and microphone. Another problem space domain where there exists a large variety of adversarial attacks is text. There is a category of attacks like [GLS+18; LJD+18; PDL19a] that generate adversarial text by replacing/swapping characters in the most influential words. The drawback of this approach is that the generated sentences will not continue to be adversarial if they are sent to a grammar checker before going to a classifier as input. Other attacks instead of replacing characters, replace complete words with their synonymous [JJZ+20; RDH+19; LMG+20b]. These attacks are less suspicious to human eyes, however, if the number of replaced words is not controlled properly the sentence might lose its original meaning.

3.3 Adversarial Attacks based on Deep Generative Models

Deep Generative Models (DGMs) tools are extensively used to generate synthetic data that matches the distribution of original data for a variety of use cases: augment scarce datasets, promote fairness, and share datasets with third parties while protecting sensitive information settings. The structure of tabular data poses some special challenges to this problem, including a mix of feature types and class imbalance. This is why a line of research is dedicated to developing new architectures for generating tabular data with DGMs. DGMs are used as a technique for the adversarial generation process. The benefit of DGMs is that once they are trained, they can efficiently generate perturbations for any instance, potentially accelerating adversarial hardening.

In this section, we expand more on the literature of DGMs for tabular data generation and that of using DGMs to generate adversarial examples.

3.3.1 DGMs for Tabular Data

Several approaches based on DGMs have been specifically designed to address particular challenges in generating tabular data such as mixed types of features, and imbalanced categorical data. Notable among these are GAN-based approaches like TableGAN [PMG+18], CTGAN [XSC+19], OCT-GAN [KJL+21], and IT-GAN [LHJ+21]. These methods leverage the power of GANs to model the underlying data distribution and generate synthetic samples that closely resemble real-world tabular data. Few approaches focus especially on particular domains like healthcare where privacy is important (see, e.g., [CBM+17; CCZ+17]). Following privacy concerns, two approaches, i.e., DPGAN [XLW+18] and PATE-GAN [JYvdS19], incorporate differential privacy techniques to ensure that the generated synthetic data does not reveal sensitive information about the individuals in the original dataset. Alternativly to GANs, Xu et al. [XSC+19] proposed TVAE as a variation of the standard Variational AutoEncoder, while TabDDPM [KBR+23] and STaSy [KLP23] were proposed following the achievements of score-based models. Finally, Liu et al. [LQB+22] proposed GOGGLE, a model that uses graph learning to infer relational structure from the data.

3.3.2 DGMs as an attack strategy

In pioneering work by Xiao et al. [XLZ$^+$18], Generative Adversarial Networks (GANs) were employed to generate adversarial examples for images. Their approach utilized the original examples as input to the generator, with the output representing the perturbation added to the original image. The generator was trained using a combination of adversarial loss, obtained by evaluating the perturbed instances on the target model, and GAN loss based on the discriminator's predictions.

Subsequent research has aimed at refining the architectures and methodologies for generating adversarial examples. For instance, Jandial et al. [JMV$^+$19] utilized the latent representations of images as input to the generator, considering them more prone to adversarial perturbations. Meanwhile, Bai et al. [BZZ$^+$21] introduced a novel attacker in the loop to train the discriminator adversarially, enhancing the attack capabilities of the generator. By employing a discriminator with two branches—one for discriminating between clean and perturbed images, and another for classifying perturbed images correctly—they proposed adversarial training of the classification module to further enhance attack ability. In another approach, Ding et al. [DTL21] explored the use of Variational Autoencoder-GAN (VAE-GAN) and concatenated original images with noise vectors as input to the generator.

Moreover, Song et al. [SSK$^+$18] focused on GAN-based adversarial examples not restricted by Lp norms. By conditioning Adversarial-Conditional GANs (AC-GANs) on desired classes, they demonstrated the ability to find misclassified examples by searching over the latent code of the generative model, highlighting the potential security threats posed by inputs that fool classifiers without confusing humans.

Extending beyond the image domain, DGM-based adversarial methods have been applied to malware and intrusion detection tasks [ZLW$^+$21; UAL$^+$19; LSX22; HT22] using similar architectures as those for images. Challenges arise from ensuring the functionality of generated examples, with researchers addressing this by preserving non-functional features, thus reducing the search space for adversarial examples.

3.4 Summary

There is a wide range of applicability for adversarial attacks beyond the computer vision from where they initially gained their popularity. Research has been carried to adversarially test these domains, often borrowing techniques from computer vision. However, it often neglects to consider the injection of domain knowledge into the adversarial generation process, resulting in unrealistic examples. Even when domain constraints are considered, they are often limited to a small subset that does not guarantee validity. And no work so far has studied the concept of realism as a problem of its own.

Table 3.1: Prior work on feature space adversarial attacks that consider domain constraints

	Year	**Use case**	**Threat**	**Method**	**Constraint type**	**Expansion to new domains**	**Code**
[KHS+18]	2018	Twitter bot Website-fingerprinting	White-box Black box	Graphical framework	Mutability Range	Define the transformation graph	Yes
[CO19]	2019	Botnet detection Malicious domains	White-box	Iterative Gradient Based	Mutability, Range, Ratio, One-hot encoding Statistical, Linear, Non-linear	Rewrite most of the components	Yes
[GCG+20]	2020	Credit scoring	Grey-box	Genetic Algorithm	Mutability, Range, Ratio, One-hot-encoding, statistical, linear, non-linear	Define constraints as a minimization problem	Yes
[LLY+20]	2020	Water treatment Power Grids	White-box Black-box	Best-Effort Search	Mutability, range Linear, Simple non-linear	Define constraints	No
[LMK+20]	2020	Price of lodging (AirBnb) Medical data ICU Credit risk	White-box	Extended JSMA	Mutability, Range,	Yes	No
[SPW+20]	2020	NIDS	White-box	Constrained Augmented JSMA	Mutability, Range, Boolean	Learn constraints from data	No
[TPS20]	2020	NIDS	White-box	Extended CW attack	Mutability	Define feature groups	No
[TWT+20]	2020	NIDS	White-box	Constraint-based IFGSM	Linear Addition boundary constraint, Zero multiplication constraint	Define constraints by visual exploration	No
[CAF+21]	2021	Fraud	Black-box	Modified Zoo, Boundary attack HopSkipJump	Mutability, Data type	Define feature weights	No
[SHP+21]	2021	NIDS Phishing	White-box	Constrained Saliency Projection	Boolean constraints	Learn constraints through Valiant's algorithm	No

4

Realism of adversarial attacks for adversarial hardening

Realistic adversarial examples are computationally expensive, hardening models with them is even more challenging. In this chapter we study the importance of realistic adversarial examples on robustifying machine learning models and pose the question whether we could improve model robustness against them using afordable non-realistic examples.

Contents

4.1 Introduction

In this chapter, we address the first challenge of this thesis, which is the high cost of adversarial hardening with realistic examples. As a reminder, many attacks that guarantee realistic examples, often come with an increased computational cost compared to traditional attacks. This additional cost can be so high that it prevents the number of examples that ML engineers can use to assess and improve robustness.

In face of this dilemma between realism and computational cost, we pose the question whether we could improve model robustness against realistic examples through adversarial hardening on non-realistic examples. A positive answer would enable model hardening at affordable computational cost and without developing specific attacks that work in the particular subject domain. We study this question through an empirical study that spans over three domains where realistic attacks exist: *text classification* where a problem-space attack (*TextFooler* [JJZ+20]) replaces words in a sentence with their synonyms; *botnet traffic detection* where a constrained feature space attack (*FENCE* [CO19]) applies realistic modifications to network traffic features; and *malware classification* where a problem-space attack (*AIMED* [CSD19]) modifies malware PE files. For each of these three domains, we generate examples using unrealistic attacks that are either domain-specific (e.g. DeepWordBug [GLS+18] for text) or generic (e.g. PGD [MMS+17]). We, then, harden the corresponding ML model on these examples and assess the resulting robustness against realistic attacks.

Our results reveal that, in the botnet use case, adversarial hardening with unrealistic examples can effectively protect against realistic attacks. But in the other use cases, unrealistic examples bring limited to no benefit, regardless of the number of unrealistic examples used for hardening. To explain these apparent discrepancies, we analyze the properties of all produced examples, in particular their location in the embedded space, their direction, their aggressiveness and their perturbed features. Through our analysis, we reveal that unrealistic attacks can be partially or completely useful if they follow the same directions as realistic attacks and have the same aggressiveness levels, as in the text and botnet case. By contrast in the malware case, the unrealistic examples follow different directions and are considerably less aggressive compared to realistic examples, which explains their inability to protect against realistic attacks.

To summarize, the contributions of this chapter are:

- Based on our literature review on realistic adversarial attacks (see Chapter 3), covering both problem-space and feature-space attacks, we select three use cases of realistic adversarial attacks and conduct the first study on adversarial hardening using unrealistic attacks that span over multiple domains.
- We reveal general insights that future research can explore to reduce the

gap between unrealistic and realistic adversarial attacks, and support model hardening against real-world adversarial threats.

4.2 Objectives and Research Questions

All the hardening methods mentioned in Chapter 2 rely on the generation of adversarial examples in order to train models that correctly classify these examples. The application of these methods to protect models against real-world adversarial attacks would, ideally, necessitate the generation of realistic adversarial examples. Realistic adversarial methods – be they constrained feature space attacks or problem space attacks – are, however, computationally expensive and specifically designed for the subject domain and problem. Table 4.1 shows the time the attacks in this study require generating a single adversarial example. Realistic attacks are 3.8 to 22650 times slower than unrealistic attacks. This largely inhibits the practical use of adversarial hardening methods, which generally require a large number of examples to protect the model effectively.

Table 4.1: Average generation time for the attacks considered in this study and the relative increase compared to unrealistic attacks.

Use-case	**Attack**	**Time (s)**	**Relative**
Text	DeepWordBug	0.15	1.0
	TextFooler (Realistic)	0.70	4.7
	PWWS (Realistic)	0.57	3.8
Botnet	PGD	0.12	1.0
	FENCE (Realistic)	0.95	7.9
Malware	PGD	0.002	1.0
	MoEvA2	36.00	18000
	AIMED (Realistic)	45.30	22650

In addition to runtime costs, one should consider the engineering effort behind the generation of adversarial examples. We identify two components in this case: the design cost to create new attacks and the engineering costs to apply them. The cost of designing new attacks is closely related to the level of realism expected as an outcome. The mapping from problem space to feature space is generally not invertible, hence many realistic attacks need to work in problem space. As Pierazzi et al. [PPC$^+$20] point out, such problem-space attacks necessitate the sequential application of problem-space (domain-specific) transformations that alter original problem-space objects into valid objects that the model misclassifies. To be realistic, these transformations must satisfy a series of conditions including: being applicable by the attacker, preserving (to some extent) the semantics of original objects, being plausible (robust to human analysis) and being robust to preprocessing. These conditions together make the engineering of the transformations challenging. Furthermore, due to the not invertible mapping, the transformations can only be applied in black-box and do not benefit from internal model feedback to drive the transformations. Ultimately, the development and assessment of problem-space transformations can only be achieved through intense experimentations. This is why the literature has primarily focused on generic, unconstrained feature-space attacks and why problem-space attacks are (although rarely) developed in specialized fields of research. Developing realistic attacks when they do not already exist is a cumbersome process.

Once these attacks are designed, they differ as well on the engineering to apply them. Some attacks like PGD are already part of standard libraries. In this case, the only costs foreseen are related to data preprocessing. Generic constrained attacks like MoEvA2 require their users to write constraints in analytical form, which typically requires the involvement of a domain expert. And lastly, there are problem-space attacks for critical domains that require particular security precautions. As an example, attacks against malware detectors have to be operated within an isolated sandbox (virtual machine) because the attacker risks corrupting its own system.

4.2.1 Research Questions

Our research investigates whether methods that generate unrealistic examples – and do so at lesser computational cost – can be used to harden models against realistic attacks.

RQ1: Can unrealistic adversarial hardening improve model robustness to realistic attacks?

We study this question across three use cases (cf. Sections 4.3, 4.4, and 4.5) that span over three domains: text classification, botnet traffic detection, and malware detection. In each of these use cases, we consider a realistic adversarial attack (a problem space attack for text and malware, and a constrained feature space attack for botnet) and at least one unrealistic attack. We apply adversarial hardening using the unrealistic attack(s) and compute the prediction accuracy of the resulting model against adversarial examples generated by the realistic attack. We compare this robust accuracy against the robust accuracy of the model hardened through realistic generation methods. In each of our experiments, we allow each method (both realistic and unrealistic) to generate the same number of adversarial examples. A gap between the two accuracy values would indicate that unrealistic methods cannot harden models as much as realistic methods enable it.

We, next, investigate whether increasing the computational budget of unrealistic model hardening (measured as the number of examples generated) can improve robust accuracy. Indeed, while unrealistic methods are computationally cheaper than realistic methods, the generation of additional examples might further help model hardening.

RQ2: Does generating more unrealistic adversarial examples help further hardening models against realistic attacks?

To answer this question, we study the trend of robust accuracy when we increase the number of adversarial examples used for hardening. A monotonic trend would motivate research on increasing the efficiency of unrealistic adversarial attacks, whereas an asymptotic behavior would indicate that unrealistic methods may never reach the effectiveness of realistic methods.

After observing the robust accuracy, our third and last research question investigates the difference between the realistic and unrealistic examples that explain the obtained results.

RQ3: What properties of the generated examples affect the hardening results?

We study, more precisely, where the examples generated by realistic and unrealistic attacks are located compared to clean examples and to each other. We do this initially through t-SNE embeddings as a means of visualizations. We expand further our study to go beyond visualizations and combine t-SNE with quantitative metrics like cosine similarity and aggressiveness (defined in Section 4.2.2). We use these quantitative metrics to highlight any insights related to the position of adversarial examples. We conclude by analyzing the distribution of changed features for each feature-space attack (Malware and Botnet use cases).

4.2.2 Metrics used

1. **Clean and Robust accuracy.** We use the term clean accuracy to refer to the standard test accuracy. We use the term robust accuracy to refer to the ratio between the number of correctly predicted adversarial examples over the total number of adversarial examples.

2. **t-SNE visualization.** We first reduce data dimensions to 30 with PCA, considering that t-SNE with high dimensional data is computationally expensive. Later we use the PCA output to generate 2-dimensional data with t-SNE and plot the results. We vary the perplexity values (10,20,30,40,50) and run the experiments with 5 different random states. One plot for each use case is shown in the paper, and a comment on how the hyperparameters affected the results are added in the respective sections.

3. **Intra-cluster dispersion scores.** We calculate the intra-cluster dispersion score as the sum of distances from adversarial examples to the centroid of adversarial examples generated by the same attack.

4. **Cosine similarity** measures the similarity between two vectors' directions. For each example, we compute the cosine similarity between the vectors formed by the application of a realistic and an unrealistic attack. We, then, average the cosine similarity over all original examples. Since cosine similarity always falls between -1 and 1, we can compare this metric across the different use cases.

5. **Aggressiveness** is the metric we introduce in order to compare how aggressively each attack works, i.e. how far the attack creates adversarial examples from the original inputs (this is the opposite notion of attack confidence [PPC+20] or adversarial friendliness [ZXH+20] that previous research has introduced). Let $X^* \subset X$ such that $x \in X^*$ if $y = t$ and $H(x) = t$. Then $x^{nn} = \underset{x^* \in X^*}{argmin} \quad D(x, x^*)$. We can now define the "*aggressiveness*" as in Equation 4.1.

$$aggressiveness = \frac{D(x, \hat{x})}{D(x, x^{nn})} \tag{4.1}$$

   Traditional distance metrics like Euclidean distance are unfortunately not uniformized over their data space, which makes it difficult to compare these distances across multiple use cases. Therefore, we divide the Euclidean distance between the initial example (x) to its adversarial counterpart ($\hat{x}$) by the distance between x and the closest original example (x^{nn}) that is predicted correctly in the target class of the attack. The distance to x^{nn} serves as a reference point for the minimum known distance to cross the decision boundary. At the same time, it allows us to have a standardized metric that can be applied to three use cases and compare between them. Having defined this metric, we compute, for each attack, the mean aggressiveness over all original examples.

We calculate the above metrics using the feature space representation of the examples (botnet, malware) and when this is not possible (text), we use the sentence embeddings calculated from the models' last hidden layer.

4.2.3 Use Case Selection

To conduct our experiments, we selected application domains and learning tasks where: (1) inputs are inherently constrained, (2) open-source datasets are available (3) realistic attacks have been proposed with available implementations. We identified many domains with datasets and constraints that originate from the problem space itself (and these constraints also translate into the feature space) and from the way features are engineered. For instance, in the botnet use case, the feature engineering results in features (a) the number of bytes, (b) the number of connections, and (c) the number of bytes per connection. All inputs must therefore satisfy the new constraints a = b * c to be realistic. This identification phase yielded a set of papers that we present in Chapter 3. However, we have observed three concerns that narrowed down our choice:

- There are only a few domains where realistic attacks (either in the problem space or the feature space) have been designed, and even fewer with a publicly available implementation. For some domains, problem space or constrained feature space attacks exist but are not entirely realistic because they produce partially valid domain objects or consider only a subset of all constraints that exist.
- An alternative to using previously proposed attacks is to tailor a generic constrained attack framework (e.g. [CO19; GCG+20]) with constraints specific to the target domain. This requires, however, either detailed documentation of the dataset features (which often overlooks the definition of constraints) or rare domain expertise that we may not have access to.
- Even when they exist, realistic attacks can have a high computational cost because they either have to resolve constraints or perform complex transformations on domain objects. Therefore, when we have several alternatives for a given domain, we tend to choose the fastest realistic attack.

Our selection process resulted in three pairs of domains/tasks and attacks that we analyze hereafter. The first pair is *text classification* and the *TextFooler* attack. This problem-space attack replaces words with synonyms and makes sure that sentences remain syntactically and semantically consistent. The second is *botnet traffic detection* and the *FENCE* attack. FENCE is a constrained feature space attack that generates realistic features of network traffic flows. The third is *Windows malware detection* and the *AIMED* attack. AIMED is a problem-space attack based on a genetic algorithm that alters the PE file of a malware without changing the malicious behavior. For each of these use cases, we conduct our experiments using state-of-the-art models and established datasets. We, moreover, rely on the model hardening processes that have originally been used to counter the realistic attacks we consider.

4.3 Text Classification

The first use case that we consider is natural language processing, in particular text classification. Morris et al. [MLL+20b] define four types of constraints for a successful and realistic adversarial example in natural language tasks: *semantics, grammaticality, overlap,* and *non-suspicion*. Semantics ensures that the adversarial transformations do not change the meaning of the sentence. Grammaticality requires that the adversarial examples do not introduce any grammatical errors. This is essential for an attack to be effective, since examples with grammar errors can easily be detected by a grammar checker. For example, using a verb tense of the future for an event happening in the present. Overlap constraints restrict the similarity between original and generated texts at the character level. Finally, non-suspicion comprises context-specific constraints that ensure the adversarial text does not look suspicious. For example, replacing a word in a modern English text with a synonym from the English of Shakespeare in an academic paper would raise suspicions. Hence, we consider as realistic any problem space attack that satisfies Morris et al.'s four types of constraints, and as unrealistic any attack that does not satisfy any of those constraints. In this use case, feature space attacks cannot guarantee the realism of the produced examples, as the transformations from text to feature space are generally not invertible.

4.3.1 Experimental Protocol

Realistic adversarial attacks (to protect against). Attacks based on synonym replacement respect all four constraints described above [MLL+20b] and are, therefore, considered realistic. We use two such attacks: TextFooler [JJZ+20], and PWWS [RDH+19]. TextFooler ranks the words based on their importance and replaces top-ranked words with semantically similar words. For example, from the sentence "*poorly executed comedy*" TextFooler could produce the adversarial sentence "***faintly** executed comedy*". PWWS uses word saliency and prediction probability to decide the word to be substituted and the replacement order for the synonyms.

Unrealistic adversarial attack (for hardening). We use the problem-space attack named DeepWordBug [GLS+18], which is based on character substitution. DeepWordBug selects the words with the highest influence on the model's decision, then applies to them character substitution transformations. For example, from the sentence "*poorly executed comedy*" DeepWordBug could produce the adversarial sentence "*p**K**orly executed comedy*". Character-level substitution obviously does not satisfy all four realism constraints and, therefore, DeepWordBug adversarial examples are unrealistic.

We evaluated the realism of the DeepWordBug and TextFooler adversarial examples by correcting their spelling with the LanguageTools proofreading service. This step filtered 87.4% of the successful DeepWordBug adversarial examples, compared to only 7.31% for TextFooler. This confirms our claim that DeepWordBug examples are unrealistic and easily detectable by language checkers.

Datasets and Model. We use Rotten Tomatoes, Tweet Offensive and AG

News datasets to train for 5 epochs a DistilBert model. You can find more details about the datasets and models in Appendix 8.1.1.

Hardening strategy. For hardening the models, we use one clean epoch at the beginning of the training process, followed up with four epochs where the training samples are the adversarial examples generated by either DeepWordBug or TextFooler. Hence, this is a case of adversarial fine-tuning with five epochs (1 clean epoch + 4 adversarial). We use TextAttack v0.2.15 library by [MLY+20] for our experiments and keep their default parameters for DeepWordBug and TextFooler when performing fine-tuning.

Evaluation. For each fine-tuned model, we evaluate the clean and robust accuracy against TextFooler and PWWS. Again, we use the default hyperparameters from TextAttack library for the evaluation. For validation set, we use the default split provided by TextAttack for Rotten Tomatoes (1066 samples) and Tweet Offensive (1324 samples). For AG News, we select randomly 1000 samples.

4.3.2 Results – RQ1 (Hardening)

In Table 4.2, we compare the robust accuracy of the models against the realistic attacks TextFooler and PWWS, of the clean model, the model fine-tuned with DeepWordBug, and the model fine-tuned with TextFoolder.

We, first, observe that, although DeepWordBug manages to increase robust accuracy in all cases compared to the clean model, it does so with significantly less effectiveness than TextFooler in all cases. For example, DeepWordBug increases the robust accuracy on PWWS adversarial sentences by +7.35% (from 12.22% to 19.57%) for Rotten Tomatoes, +9.56% for Tweet Offensive, and +7.5% for AG News. By contrast, fine-tuning with TextFooler improves the robustness against PWWS by 15.29%, 26.5%, and 13.35%, respectively. Similar observations are made when TextFooler is the attack to protect against.

Interestingly, TextFooler manages to harden the model against PWWS as well as it does against TextFooler itself. This indicates that, in this text classification use case, the protection brought by a given realistic attack generalizes to other realistic attacks. Though unrealistic attacks like DeepWordBug have a similar effect across the two realistic attacks, their benefits are far from those obtained with a realistic attack.

Note that fine-tuning with any of the two attacks has a negligible impact on the clean performance of the model, indicating that adversarial hardening, in this case, does not hurt performance against benign sentences.

4.3.3 Results – RQ2 (Budget)

We investigate if generating more adversarial examples can help DeepWordBug achieve the same hardening effectiveness as TextFooler. We repeat our experiments with DeepWordBug for up to 20 epochs (instead of 5 initially), and we record

Table 4.2: Clean and robust accuracy (in %) of the clean model and the adversarially fine-tuned models, over three datasets. The first column represents the clean accuracy, and the remaining columns represent the robust accuracy against TextFooler and PWWS.

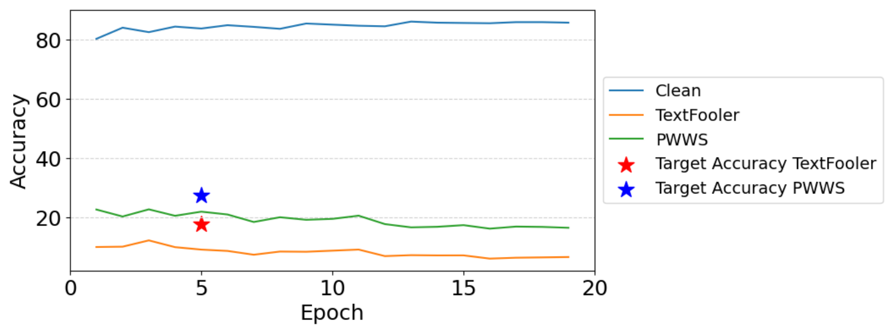
Dataset	**Trained against**	**Attacked by**		
		-	TextFooler	PWWS
Rotten tomatoes	Clean	83.68	5.49	12.22
	DeepWordBug	83.40	9.34	19.57
	TextFooler	80.49	**17.72**	**27.51**
Tweet Offensive	Clean	81.04	11.56	18.45
	DeepWordBug	77.11	19.20	28.01
	TextFooler	78.47	**43.21**	**44.95**
AG News	Clean	89.60	16.07	32.37
	DeepWordBug	89.80	17.29	39.87
	TextFooler	89.90	**26.14**	**45.72**

for each epoch the robust accuracy against adversarial examples generated by TextFooler and PWWS. The increased number of epochs from 5 to 20 means we are multiplying the number of generated examples by 4, as we generate one example per original input in each epoch.

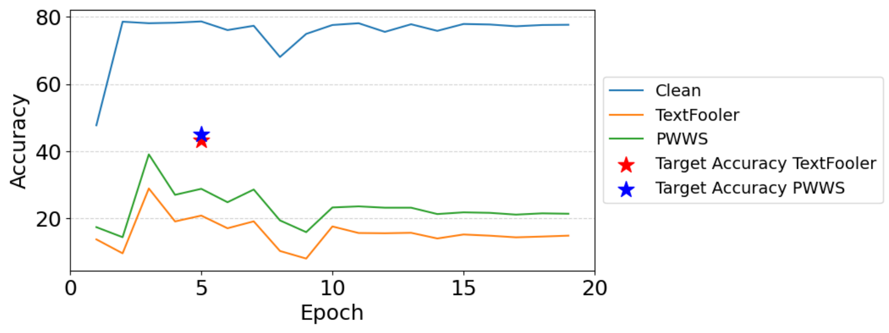
We show the results for the three datasets in Figure 4.1. Though the robust accuracy values do fluctuate between the first and 15th epochs, the robustness that DeepWordBug-based hardening achieves never reaches the same level as the robustness previously achieved by hardening with TextFooler. Compared to our initial number of 5 epochs, increasing the number of epochs does not bring positive effects. This might indicate that the unrealistic attack generates examples confined to a limited space, and that a significant number of realistic adversarial examples lie outside this space. We investigate this further in RQ3.

4.3.4 Results – RQ3 (Properties)

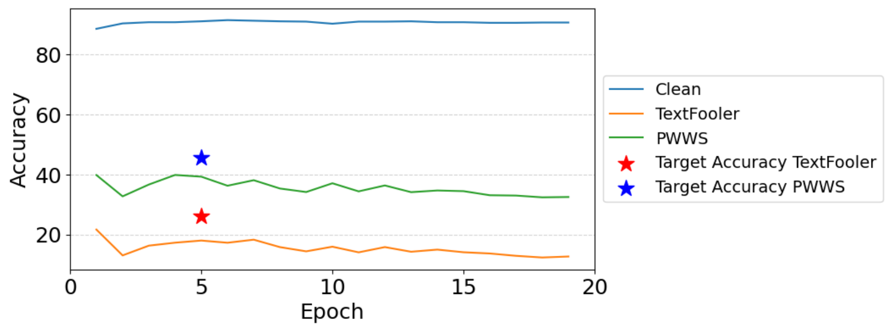
Following our results above, we hypothesize that the adversarial examples generated by TextFooler target different areas that the DeepWordBug adversarial examples do not reach. To facilitate the investigation of this hypothesis, we focus on Rotten Tomatoes and Tweet Offensive as these datasets are binary (AG News is multi-class). We analyze the sentence embeddings obtained from the last layer latent representation of the clean model.



(a) Rotten Tomatoes



(b) Tweet Offensive



(c) AG News

Figure 4.1: Robust accuracy of models adversarially fine-tuned with DeepWordBug over 20 epochs for the three datasets. Red line is robust accuracy against TextFooler, green line is against PWWS, blue line is clean accuracy. Stars represent the robust accuracy achieved by a 5-epoch TextFooler-based fine-tuning against TextFooler (red star) and PWWS (green star).

Embeddings of the adversarial examples To compare the embeddings of the different clean and adversarial examples, in Figure 4.2, we visualize the t-SNE embeddings for the positive reviews in Rotten Tomatoes. Varying the value of the perplexity and using four other random states did not impact majorly the general formed structures. The hyperparameters affected only the graph stretching or graph rotations. We see that all adversarial examples generally lie between the two main class clusters (orange and blue dots), indicating that the attacks indeed navigate around the decision boundary. However, the DeepWordBug examples are closely grouped together, with a dense concentration at specific areas. Meanwhile, TextFooler examples have a wider spread throughout the decision boundary. The rest of the plots for Rotten Tomatoes and Tweet Offensive are shown in Appendix 8.1.2 and corroborate our findings from Rotten Tomatoes.

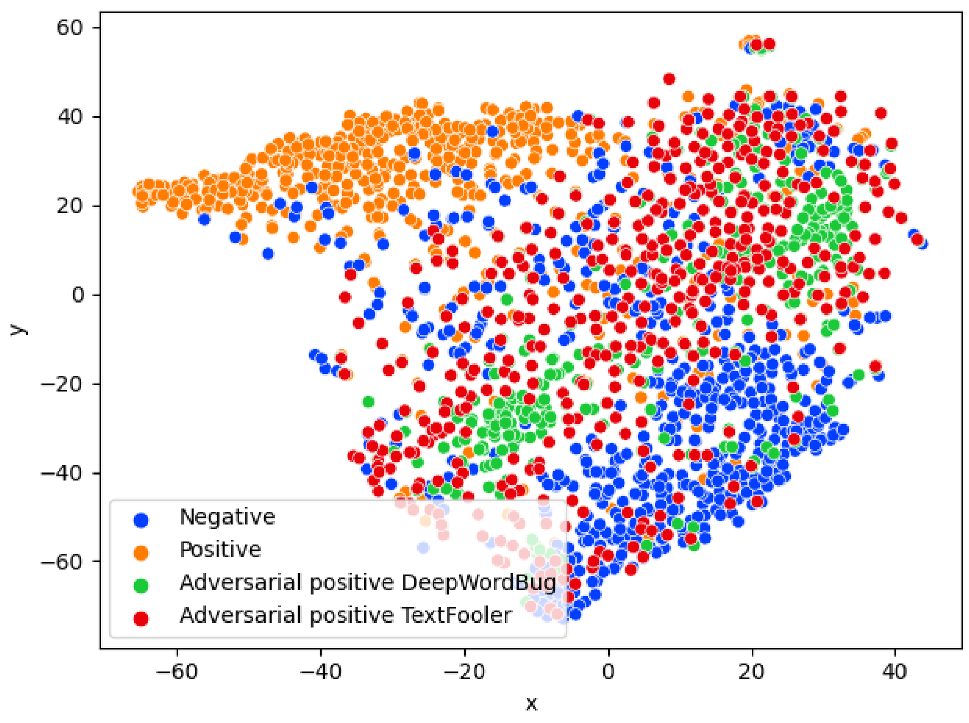


Figure 4.2: t-SNE visualization of original examples and the successful adversarial examples generated by DeepWordBug and TextFooler for Rotten Tomatoes positive reviews.

This qualitative analysis confirms our hypothesis that TextFooler examples cover a larger portion of the space that the DeepWordBug examples never reach. This is the reason why DeepWordBug offers only limited improvement against realistic attacks.

Clustering analysis of the adversarial examples To complement our analysis, we present in Table 4.3 the intra-cluster dispersion score of the adversarial examples generated by DeepWordBug and TextFooler before applying the dimension reduction with t-SNE. A higher intra-cluster dispersion score indicates that the cluster is more spread over the latent space. This quantitative analysis confirms that adversarial examples generated by TextFooler cover more of the latent space. For instance, on the Rotten Tomatoes dataset, the TextFooler adversarial examples produced from positive original examples have an intra-cluster dispersion score 71.7% higher than the DeepWordBug adversarial examples produced from the same original examples.

Table 4.3: Intra-cluster dispersion scores of adversarial examples generated by DeepWordBug and TextFooler for each class. A higher score indicates that the cluster is more spread.

Dataset	**Class**	**DeepWordBug**	**TextFooler**
Rotten Tomatoes	Negative	173899	241840
	Positive	155892	267629
Tweet Offensive	Offensive	221589	452303
	Non-offensive	379399	401270

Perturbation size and direction Table 4.4 gives the mean and standard deviation values for cosine similarity and aggressiveness of adversarial examples. For both datasets, DeepWordBug and TextFooler generate adversarial examples that follow very similar directions, considering that their cosine similarities have high values, respectively 0.89 for Rotten Tomatoes and 0.92 for Tweet Offensive. Additionally, they are similarly aggressive, with their mean aggressiveness score differing only by 0.06 for Rotten Tomatoes and 0.02 for Tweet Offensive. This explains why DeepWordBug is still able to partially robustify the models that are faced with TextFooler examples.

4.3.5 Conclusion for Text Classification

Our evaluation confirms that unrealistic adversarial hardening does not suffice to protect against realistic adversarial attacks. Unrealistic adversarial hardening leads to models up to 16.94% less robust than realistic adversarial hardening against real attacks. We also demonstrate that using more unrealistic adversarial examples in the hardening (up to four times more) does not lead to any further improvement of the robustness of the models against realistic attacks. We justify that we cannot fully replace realistic adversarial examples with unrealistic because unrealistic adversarial examples do not cover the feature space as well as the

Table 4.4: Mean and standard deviation values for cosine similarity and aggressiveness of adversarial examples.

Dataset	**Attack**	**Cosine similarity**		**Aggressiveness**	
		Mean	**Std**	**Mean**	**Std**
Rotten Tomatoes	DeepWordBug	0.89	0.10	0.90	0.24
	TextFooler	1.00	0.00	0.84	0.27
Tweet Offensive	DeepWordBug	0.92	0.09	0.78	0.34
	TextFooler	1.00	0.00	0.80	0.37

realistic examples. We demonstrate how realistic examples are widely spread, less clustered and better cover both classes of the binary classification task than unrealistic adversarial examples. However, the unrealistic examples have similar direction and aggressiveness level as realistic examples, hence they still manage to provide some level of robustification.

4.4 Botnet Traffic Detection

The detection of botnet traffic is a successful application case of machine learning [LWL+06; TLS+20]. The resulting detectors typically learn from the network flows between the source and destination IP addresses that botnet and normal traffic generate. Network traffic is naturally subject to validity constraints. For example, the TCP and UDP packet sizes have a minimum byte size, and a change in the number of packets sent always brings a change in the total bytes sent. We are unaware of any problem space attacks for botnet detection. We, therefore, consider a constrained feature space, FENCE [CO19], that claims to be realistic. The FENCE framework guarantees that it produces adversarial examples in feasible regions of the feature space, therefore they can be projected back to the raw input space. The projection operator needs to be defined for each application domain separately.

4.4.1 Experimental Protocol

Realistic adversarial attack (to protect against). We use the FENCE attack designed by [CO19]. FENCE is initially a generic framework that has been tailored to botnet traffic detection, and we reuse the original implementation presented in [CO19]. FENCE is a gradient-based attack that includes modifications such as mathematical constraints and projection operators to create feasible adversarial examples. As denoted in the original paper, a family of features is a set of features related by one or more constraints. The algorithm starts by finding the feature of maximum gradient, referred as the representative feature, and its respective family. The representative feature of the family is updated with a value δ, and the other members are updated based on δ and the constraints definitions. The value δ is found through a binary search such that the global perturbation does not exceed the maximum distance threshold. Therefore, the constraints are satisfied by construction.

Unrealistic adversarial attack (for hardening). We use the PGD implementation from ART [NST+18]. PGD is a traditional iterative gradient attack that adds a perturbation based on gradient direction at every iteration. We modify the implementation from ART to stop PGD updates at the moment an adversarial example is found. This modification is equivalent to Friendly Adversarial Training presented by [ZXH+20] which does not degrade the clean accuracy of the models.

As a preliminary check, we have evaluated the attacks' level of realism using the analytical feature constraints derived from Simonetto et al. for botnet datasets. Respecting these constraints is a necessary but not sufficient condition for having realistic examples, since the problem space is more restricted than the feature space. Failure to satisfy them indicates unrealistic examples. None of PGD examples respect the constraints and 100% of FENCE examples respect the constraints. In addition, by design, it is always possible to project FENCE adversarial examples to their raw input space. This confirms that PGD attack is unrealistic and FENCE attack is realistic.

Datasets and Model. We use Neris, Rbot and Virut from standard CTU-13 dataset to train a neural network architecture with dense layers from the original FENCE paper [CO19] for 10 epochs. You can find more about the datasets in Appendix 8.1.1.

Hardening strategy. We harden the model performing adversarial training for 10 epochs. For both attacks, we use 5 iterations to generate the adversarial examples and a perturbation budget of $\epsilon = 12$ under a L_2 norm. The L_2 distance is the one from the original FENCE paper [CO19].

Evaluation. For evaluation, we use 100 iterations for both PGD and FENCE. In the botnet domain, the adversary is interested on passing a flow as normal traffic and not vice-versa. Therefore, we apply adversarial attacks that target the benign botnet class.

4.4.2 Results – RQ1 (Hardening)

Table 4.5 compares the robust accuracy (when attacked by FENCE) of the clean models, the models adversarially trained with PGD, and the models adversarially trained with FENCE. We observe that the PGD-trained model is perfectly robust against the FENCE attack. Across the three datasets, the robust accuracy jumps to 100% after adversarial training with PGD (just like with FENCE). Meanwhile, the clean performance of the models is not affected. This reveals that even the unrealistic, unconstrained attack can protect the model as effectively as the realistic attack.

Table 4.5: Clean performance (measured with ROC AUC) and robust accuracy (against FENCE applied on botnet examples) of the original botnet detection model and the adversarially hardened models.

Training scenarios		**Roc AUC**	**Attacked by FENCE**
Neris (1,9)	Clean	97.54	0.00
	PGD	97.30	100.00
	FENCE	96.61	100.00
Rbot (10,11)	Clean	63.52	84.85
	PGD	63.52	100.00
	FENCE	59.64	100.00
Virut (13)	Clean	79.96	70.84
	PGD	78.75	100.00
	FENCE	71.91	100.00

4.4.3 Results – RQ2 (Budget)

This research question is not relevant in the botnet case. Under the same budget of epochs and attack iterations, PGD-adversarial training already outperforms FENCE-adversarial training by 0.7% in the previous research question. There is no significant utility in increasing the budget of examples to generate.

4.4.4 Results – RQ3 (Properties)

Our RQ1 results demonstrate that unrealistic hardening (PGD) and realistic hardening (FENCE) yield similar robustness against problem space attacks. Based on our findings in the text classification case, we hypothesize that both attacks explore the same search space, and that real botnets lie close to this explored vulnerable space.

Embeddings of the adversarial examples. In Figure 4.3 we visualize the clean and adversarial examples that each attack generated using t-SNE on Neris dataset. We see that both PGD and FENCE generate adversarial examples targeting particular neighbourhoods, forming “islands" of adversarial examples. Varying the value of the perplexity and random state did not impact majorly the general structures formed by the normal traffic examples. However, they impacted the position of these “islands". Fence and PGD examples were always forming independent clusters from each other, however their position in the plot was not stable. They were sometimes closer and sometimes further from each other, and initial botnet examples while preserving their structure. This indicates that PGD and FENCE direct their examples to single blind-spots, however from t-SNE visualization itself we can not conclude that these spots are in close neighbourhoods.

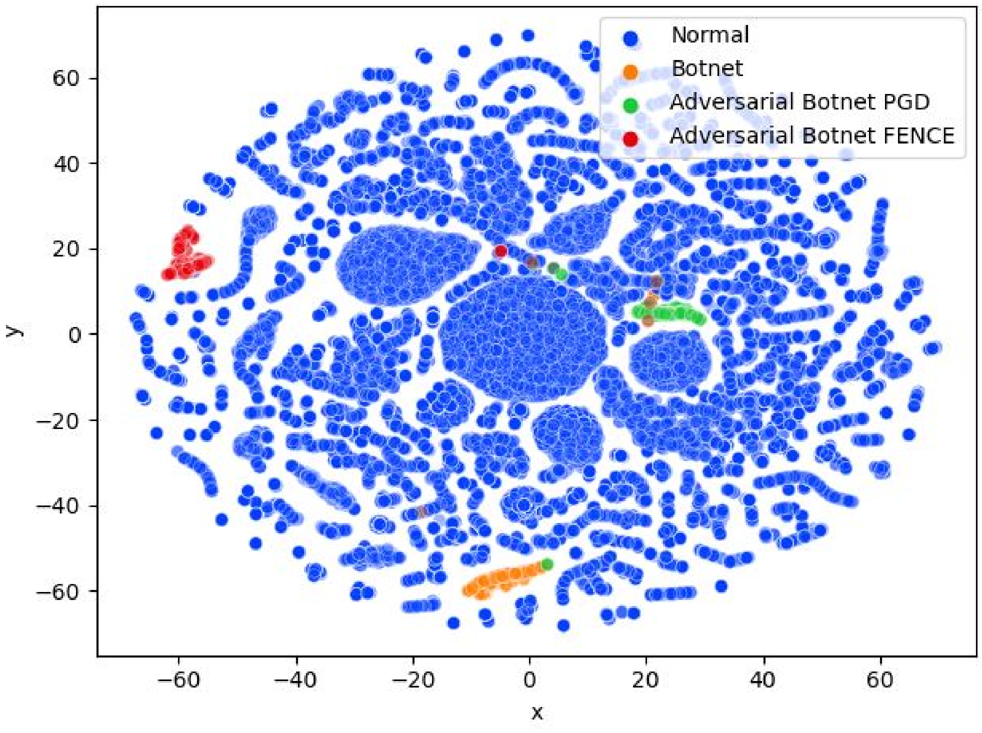


Figure 4.3: t-SNE visualization of original and successful adversarial examples generated by PGD and FENCE.

Perturbation size and direction. Table 4.6 presents cosine similarity and aggressiveness values for adversarial examples. The direction of PGD and FENCE adversarial examples is almost the same, with their mean cosine similarity very close to 1 for the three datasets, respectively 0.88 for Neris, 0.97 for Rbot and 0.90 for Virut. Moreover, they are similarly aggressive because the difference between their aggressiveness scores is low. Respectively 0.37 for Neris, 0.03 for Rbot and 0.41 for Virut. We conclude that this is one of the reasons why PGD is able to protect against FENCE examples.

As a second step, we investigate whether PGD and FENCE perturb the same features. Figures 4.4a and 4.4b show the number of examples where each feature has been perturbed by PGD and FENCE, respectively. We observe that PGD perturbs all the features that are mutable for all botnet samples. Meanwhile, FENCE has

Table 4.6: Mean and standard deviation values for cosine similarity and aggressiveness of adversarial examples.

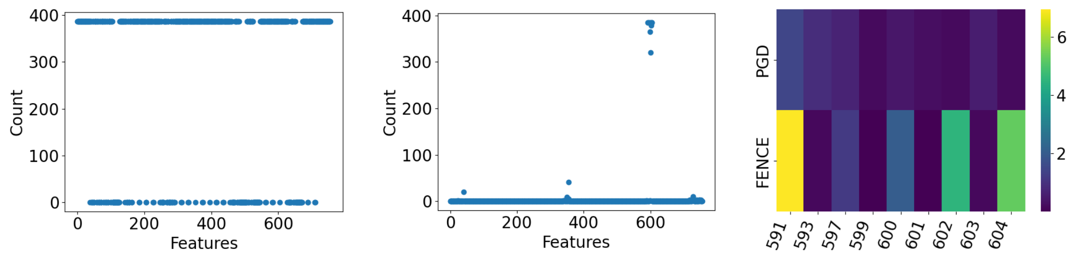
Dataset	**Attack**	**Cosine similarity**		**Aggressiveness**	
		Mean	**Std**	**Mean**	**Std**
Neris	PGD	0.88	0.05	0.12	0.05
	FENCE	1.00	0.00	0.49	0.22
Rbot	PGD	0.97	0.02	0.06	0.01
	FENCE	1.00	0.00	0.09	0.13
Virut	PGD	0.90	0.09	0.13	0.06
	FENCE	1.00	0.00	0.54	0.43

a small set of features that are perturbed in more than half of the samples, but most of them are never updated. To show the magnitude of the perturbation, in Figure 4.4c, we display the average perturbation size for features that were updated in more than 200 samples by FENCE. The heatmap shows that PGD perturbs all features almost equally and with low effect, except for feature 591 – the most perturbed. Feature 591 is also the feature that FENCE perturbs the most, and one of the only two that FENCE significantly perturbs. This means that perturbing feature 591 is the main factor to make an example reach the non-botnet class, and the two attacks exploit this feature to generate their adversarial examples. To confirm, we check the feature 591, and it is the feature related to the bytes sent from port 443 named *bytes_out_sum_s_443*. This port is indeed commonly exploited by botnet traffic. Hence, its values are exploited by the attacks to pass a botnet example as normal traffic.

Ultimately, we explain the success of PGD in protecting against FENCE by two facts: (1) both PGD and FENCE examples follow the same direction and are similarly aggressive, (2) both FENCE and PGD apply the highest perturbation to the same feature.

4.4.5 Conclusion for the Botnet Detection Use Case

In this section, we studied the hardening of the models that detect botnet traffic based on network flows. For this use case, an adversary is interested to “disguise" botnet traffic as normal traffic while preserving some inherited domain constraints that are related to packet size, time, protocol and ports relationship, etc. An attack that respects these constraints is the gradient based attack FENCE. We study how two different strategies of adversarial training affect the models’



(a) PGD (b) FENCE (c) Average perturbation

Figure 4.4: Analyses of features perturbed by PGD and FENCE. Subfigures a and b show, for each feature, the number of examples where the feature has been perturbed by PGD and FENCE, respectively. Subfigure c shows the average perturbation size caused by PGD and FENCE to the features that are perturbed more than 200 times by FENCE.

robustness against FENCE attack. The first strategy includes in the training process adversarial examples generated by the unconstrained attack PGD, and the second includes realistic constrained examples generated by FENCE. The results showed that despite the fact that PGD does not generate realistic examples, it was able to increase the robust accuracy of the model by 96%, similar to 95.30% by FENCE itself. The reason for this increase in robustness seems to be that the examples generated from PGD and FENCE follow the same direction and are similarly aggressive. In addition, they have similar characteristics in feature perturbation. Being able to use PGD adversarial training to defend against a constrained attack like FENCE, brings benefits to the training costs, because PGD runs much faster than FENCE.

4.5 Windows Malware Detection

Malicious software has always been a threat to the security of IT systems. With the recent surge of machine learning, research and practice have developed many learning-based detection systems that are able to distinguish between benign and malicious software [UAB19]. However, the same systems are also vulnerable to adversarial attacks that syntactically modify the malware without reducing its malicious effect [CSD19; PPC^{+}20]. These attacks typically work in the problem space, e.g. they modify the malware PE file. Pierazzi et al. [PPC^{+}20] expose the conditions for a problem-space attack to be realistic, including preserved semantics, plausibility, robustness to preprocessing. We consider one such technique – AIMED [CSD19] – as the realistic attack to protect against. For model hardening, we consider different unconstrained feature space attacks, as well as constrained feature space attacks that capture some properties and dependencies of the features but cannot guarantee that the altered features can be mapped back to a real malware.

4.5.1 Experimental Protocol

Realistic adversarial attack (to protect against). We consider AIMED, a problem-space black-box attack [CSD19]. AIMED injects perturbations to the malware PE file in order to create variants of the malware, and runs a genetic algorithm to make these variants evolve. The attack aims to optimize four components: functionality, misclassification, similarity, and generation number. As recommended by the results of the original AIMED implementation [CSD19], the number of perturbations is 10 in order to reduce invalid examples. The generated malware files are sent for execution to a sandbox, for a functionality check. The sandbox setup is the one used by [CSD19] for AIMED evaluation. If functional, statistical features are extracted and fed to the classifier, which evaluates if the perturbed malware is evasive. On average, it takes 1 - 2 minutes to process a single example of PE files.

Unrealistic adversarial attacks (for hardening). We again use the PGD implementation from ART [NST+18] as an unconstrained unrealistic attack. We also consider MoEvA2 [SDG+21a], a generic framework for constrained feature-space adversarial attacks that are based on genetic algorithms. MoEvA2 comes with a generic objective function that considers perturbation size, classification results, and constraint satisfaction. We identify immutability constraints based on the PE format description and structure given by Microsoft. We also extract feature dependency constraints from the original PE file examples we collected and those generated by AIMED. For example, the sum of binary features set to 1 that describe API imports should be less than the value of features `api_nb`, which represents the total number of imports on the PE file. In our experiments, we use MoEvA2 as described above, as well as an alternative version where we remove the constraints in order to simulate an unconstrained attack. We will refer to this version as *U-MoEvA2*.

As a preliminary check, similarly to the botnet use case, we have evaluated the attacks' level of realism using the analytical feature constraints derived from Simonetto et al. for malware use case. As a reminder, respecting these constraints is a necessary but not sufficient condition for having realistic examples, since the problem space is more restricted than the feature space. Failure to satisfy the contstraints indicates unrealistic examples. For this use case, respectively 0% of PGD, 2% of U-MoEvA2, 100% of MoEvA2 and 100% of AIMED adversarial examples respect the constraints. For U-MoEvA2 and MoEvA2 there is no guarantee that we can project back to the problem space since the problem -feature space relation is not invertible. On the other hand, AIMED generates an actual problem space object (Windows PE file) that are already checked for functionality. This confirms that PGD, U-MoEvA2 and MoEvA2 attacks are unrealistic in malware scenario, meanwhile AIMED is guaranteed to be realistic.

Dataset and Model. We use a collection of benign and malware PE files provided in [AGM+20] to train a Random Forest model. You can find more details about the dataset and model in Appendix 8.1.1.

Hardening strategy. To harden the model, we adversarially retrain the model with examples generated by PGD, U-MoEvA2, MoEvA2, and AIMED and retrain the classifier. Due to the costs related to generating adversarial examples with AIMED, we randomly select a sample of 1500 original malware examples, and we use only these examples to generate adversarial examples for retraining with the four attacks.

For adversarial examples generation, we study different ranges of ϵ thresholds. PGD does not change the maximum L_2 distance of its generated adversarial examples after $\epsilon = 0.1$. Additionally, the maximum L_2 distance at the threshold of 0.1 is much lower than the maximum L_2 achieved by AIMED. A threshold of 0.1 is also enough for MoEvA2 to generate adversarial examples that were later used for retraining.

To generate adversarial examples with PGD, we train on the original training set a neural network with dense layers that achieves a ROC AUC score of 0.91, equal to the Random Forest. For the number of PGD iterations, we keep the default value in ART (100).

Evaluation. For evaluation, we use the subset of test examples where the prediction of the model and the actual label corresponds to malware (1069 for the clean model). We do not evaluate the robustness of the normal class because we consider that an attacker has no interest in our knowledge into fooling a classifier to identify a normal file as malware.

4.5.2 Results – RQ1 (Hardening)

Hardening the model with unrealistic examples does not improve the robustness of malware detection models against realistic adversarial attacks, even when the unrealistic examples respect some validity constraints. Table 4.7 shows for each model the clean accuracy and the robust accuracy against the realistic adversarial examples generated by AIMED. The clean model has an accuracy of 76.25% on AIMED examples. The model benefits from retraining only with examples generated by AIMED itself (+18.99%). Retraining with PGD gives a negligible improvement of 0.58% and retraining with MoEvA2 or U-MoEvA2 slightly decreases the robust accuracy. In addition, hardening with AIMED does not hurt clean performance – it actually improves it by 1.60%.

4.5.3 Results – RQ2 (Budget)

We, next, increase the number of adversarial examples generated by unrealistic attacks from 1,500 to 6,000 (using different original examples) and we observe the

Table 4.7: Clean and robust accuracy (against AIMED applied to the malware class) of the clean model and the adversarially retrained models.

Trained against	**Attacked by**	
	-	AIMED
Clean	90.60	76.25
PGD	90.60	76.83
MoEvA2	90.60	75.75
U-MoEvA2	90.71	75.35
AIMED	91.20	95.24

resulting robust accuracy against AIMED. Adversarial retraining with unrealistic examples does not benefit from this increased budget. Figure 4.5 shows that the robust accuracy of PGD, U-MoEvA2, and MoEvA2 is constant against realistic adversarial examples generated by AIMED.

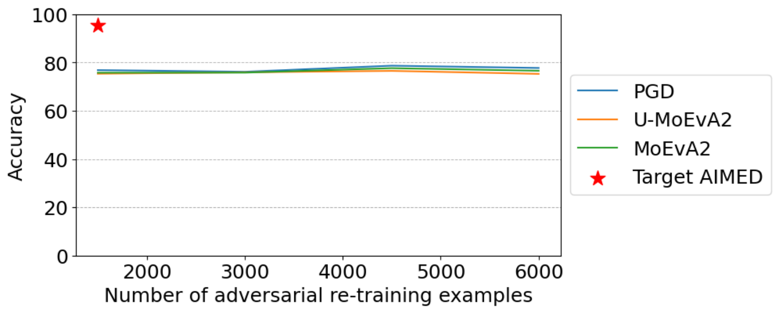


Figure 4.5: Robust accuracy against AIMED when increasing the number of adversarial retraining examples from 1500 to 6000, using each unrealistic attack. The green star represents the robust accuracy achieved retraining with 1500 adversarial examples generated by AIMED.

4.5.4 Results – RQ3 (Properties)

Our results demonstrate that feature space hardening remains inferior to problem space hardening even with additional budget. We hypothesize that problem space attacks and feature space attacks (even with constraints) do not explore the same search space, and that problem space attacks better cover the vulnerable space of malware detection models.

Embeddings of the adversarial examples In Figure 4.6 we visualize, using t-SNE, the embeddings of the clean training set and of the adversarial examples that each attack has generated. Varying the value of the perplexity and random state did not impact the major structures formed in the graph. The hyperparameters only impacted the orientation and localization of the clusters. We see that PGD examples form 2 major clusters. Similarly, examples from U-MoEvA2 form two clusters in two different neighbourhoods. Meanwhile, MoEvA2 examples do not particularly fall on the same neighbourhood. They are more spread throughout the graph. Lastly, the examples generated by AIMED cover a larger surface of the graph, with most of them following the embedding patterns of the normal files.

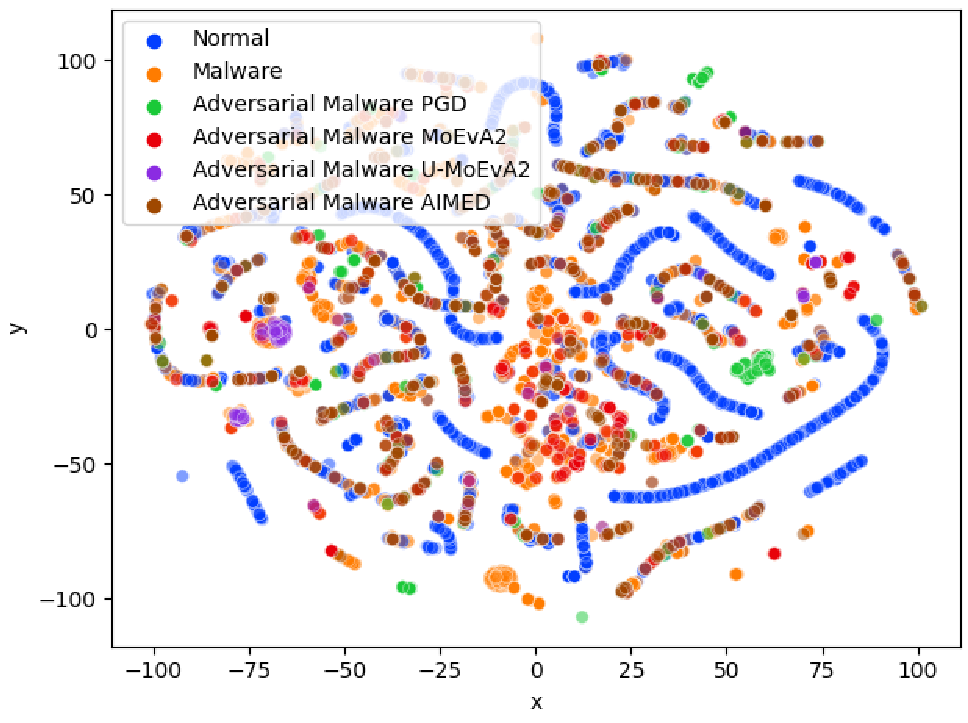


Figure 4.6: t-SNE visualization of original examples and the successful adversarial examples generated by PGD, U-MoEvA2, MoEvA2 and AIMED.

Perturbation size and direction Table 4.8 presents cosine similarity and aggressiveness values of adversarial examples. The adversarial examples generated by PGD, U-MoEvA2 and MoEvA2 follow different directions compared to AIMED examples. The mean cosine similarity to AIMED examples is equalling 0.39 for all of them. As a reminder, a cosine similarity of 0 indicates orthogonal vectors. Regarding the aggressiveness, AIMED adversarial examples are clearly more aggressive than unrealistic adversarial examples. They have a difference in mean aggressiveness value of 42.24 with PGD and MoEvA2 and 42.25 with U-MoEvA2.

These observations are clearly different from text classification and botnet detection use cases, where adversarial examples are more cosine-similar and have the same level of aggressiveness. As a result, we conclude that this explains why hardening with unrealistic examples does not robustify the model against realistic examples in this use case.

Table 4.8: Mean and standard deviation values cosine similarity and aggressiveness of adversarial examples.

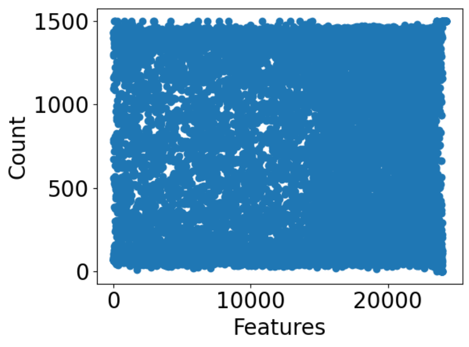
Attack	**Cosine similarity**		**Aggressiveness**	
	Mean	**Std**	**Mean**	**Std**
PGD	0.39	0.17	0.02	0.04
U-MoEvA2	0.39	0.17	0.01	0.07
MoEvA2	0.39	0.17	0.02	0.00
AIMED	1.00	0.00	42.26	629.34

In Figure 4.7 we show, for any feature f, the number of adversarial examples that have a different value of f compared to the original example it was generated from. PGD (4.7 (a)) perturbs almost all features. This is an expected behaviour as only 141 features out of 24 222 are not modifiable. The modified features are updated by PGD based on the direction of the gradient. We see in 4.7 (b) and (c) that both U-MoEvA2 and MoEvA2 perturb similarly only the last features that are related to the byte frequency. It seems that the constraints related to these features are easier to satisfy, so MoEvA2 does not explore the full search space.

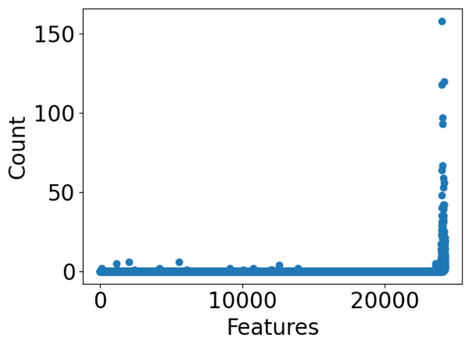
In 4.7 (d) we see that AIMED has a different perturbation pattern compared to the three previous attacks. Similarly to MoEvA2, it perturbs at the end features related to byte frequencies but as well other features in front of them. Another difference among them is that AIMED perturbs the first half of the features, which are related mostly to API imports. 350 features are perturbed more than in half examples, with 266 of them perturbed always. AIMED avoids perturbing the middle features related to *dll* imports, although these features are mutable. This is because the domain-specific transformations that AIMED implements never import *dll*.

4.5.5 Conclusion for the Malware Use Case

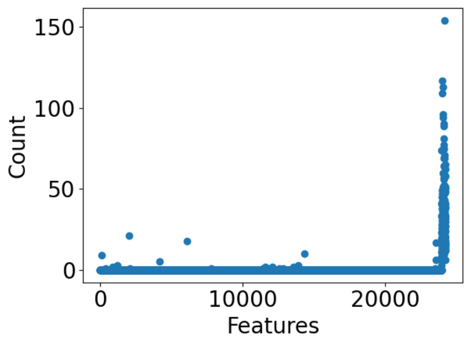
Our evaluation shows that for the malware classification use case, feature space hardening using PGD and U-MoEvA2, even when driven by domain constraints (MoEvA2) fails to harden the model. We demonstrate that even when given 4 times more budget, the absence of noticeable effect remains. Our investigation reveals that the realistic attack produces adversarial examples that follow different



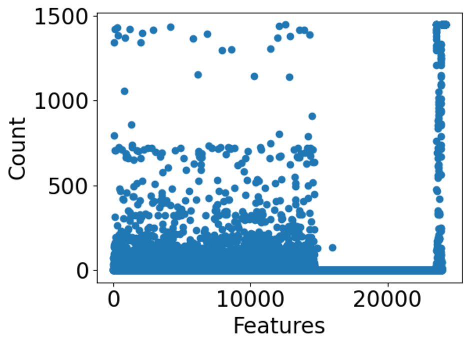
(a) PGD



(b) U-MoEvA2



(c) MoEvA2



(d) AIMED

Figure 4.7: Number of times each feature is perturbed in the generated adversarial examples used for retraining by PGD, U-MoEvA2, MoEvA2 and AIMED.

directions from the unrealistic examples. Moreover, the realistic attack is clearly more aggressive than the unrealistic attacks. The unrealistic feature-space attacks tend to modify either a small set of features or almost all features, differently from problem space attack AIMED. AIMED is capable of generating adversarial examples similar to the clean examples, which feature-space attacks are incapable of doing. Consequently, these areas of the model's decision boundaries are not explored by feature space attacks and remain vulnerable to the realistic attack.

4.6 Wrap-up and Perspectives

Since the seminal work that uncovered the vulnerability of machine learning models to adversarial examples[BCM+13], researchers and practitioners have realized the potential harm that adversarial examples could cause in the real world [PPC+20; SEZ+20]. Realistic adversarial attacks remain more challenging to design and, therefore, harder to study and protect against. In this chapter, we propose the first work in our knowledge that evaluates the limitations and opportunities of adversarial hardening against realistic adversarial examples.

Across our three use cases from three domains, we have observed that the effectiveness of adversarial hardening using unrealistic examples significantly varies: it ranges from not increasing model robustness at all (malware case) to achieving 99.70% robust accuracy (botnet case), or sometimes offering partial protection (text case). These results suggest the absence of a general trade-off between the realism of the produced examples and the potential of using these examples to protect against real-world attacks. This maintains hope that research on traditional adversarial attacks can still be useful to protect against domain-specific real-world attacks, and these domains go beyond computer vision. It is important to find a relative balance between the effort of a malicious third party to attack the system (i.e. design the domain-specific realistic attack and run it) and the effort of engineers to defend the system (i.e. through adversarial hardening). For practitioners and researchers, this suggests that *unrealistic examples may help under strict conditions, hence they are worth a try.*

In all our use cases, we have observed that increasing the number of generated unrealistic examples has negligible effects on robustness. This suggests that there is an asymptotic limit to the utility of unrealistic examples in model hardening and defending against realistic attacks effectively is not a matter of computational budget involved, but is rather a matter of the characteristics of these generated examples. Therefore, improving the efficiency of unrealistic attacks — a dedicated area of research — is worthless in these settings. Research should rather work on profoundly changing these attacks. Our takeaway is that *if unrealistic examples do not bring improvement even at a small scale, they will never do, regardless of the allocated computational budget.*

Through our in-depth analysis of all adversarial examples, we have highlighted that unrealistic examples are useful only if they have a close representation to the realistic examples (i.e. they follow the same direction, they are similarly aggressive, and they perturb the same features). Indeed, the botnet case has revealed that unrealistic examples and realistic examples overlap in the feature space, and the attacks that generated these examples typically modify the same features. *Our results pave the way for developing new adversarial hardening mechanisms. These mechanisms would rely on new attacks (or adaptation of existing attacks) to efficiently produce unrealistic examples that mimic realistic adversarial examples.* This follows the assumption that a set of realistic adversarial examples is available (e.g. captured from previous attack attempts or produced by an existing attack). As a future work, one of those considered mechanisms is learning domain constraints from realistic examples, and then integrating the constraints as an attack objective (e.g. into PGD). Another foreseen mechanism would drive the attack in a direction towards realistic examples — achieving a high cosine similarity and a similar degree of aggressiveness — in order to produce examples that are close in their latent space.

The proposed properties to explain the effect of the robustification with unrealistic or unrealistic adversarial examples (i.e aggressiveness, feature perturbation) were just a subset of metrics that we explored. Some other methods like distribution shift detectors, loss landscapes, etc. did not provide any meaningful results and were not included in this chapter. However, in general, we noticed in the literature a lack of mechanisms to understand the behaviour of models under robustification. *While we believe that advancing the state of the art with new attacks and defenses is important, a larger focus needs to be given to explainability methods as they are crucial factors in the intuition of what can work well in the future as a defense against realistic attacks.*

While this work is general in the sense that it covers three use cases each expanding in completely different domains, the settings on which we study them are fixed (i.e a single model). This limitation is not addressed in this thesis, mostly due to the already large size of the study. *However, we hope this work will inspire other researchers to conduct similar studies more in-depth for each one of these domains.*

5

Realism of adversarial examples in NLP

In NLP, adversarial attacks aim to cause failures in the model by slightly perturbing the input text in such a way that its original meaning is preserved In this chapter we argue that these attacks should produce text that is both valid and natural to be realistic and effective.

Contents

5.1 Introduction

In this chapter, we address the second challenge of this thesis: realism concept adversarial attacks in NLP, and partially the third challenge: the spectrum of realism. Like many other machine learning models, Natural Language Processing (NLP) models are susceptible to adversarial attacks. In NLP, these attacks aim to cause failures (e.g. incorrect decisions) in the model by slightly perturbing the input text in such a way that its original meaning is preserved. However there is no consensus on what is considered a realistic adversarial example in NLP.

Research has reported on the potential of adversarial attacks to affect real-world models interacting with human users, such as Google's Perspective and Facebook's fastText [LJD+19]) More generally, these attacks cover various learning tasks including classification and seq2seq (fake news [LMG+20a], toxic content [LJD+19], spam messages [KNL20]), style transfer [QCZ+21] and machine translation [MLN+19]).

It is critical to properly assess model robustness against adversarial attacks to design relevant defence mechanisms. This is why research has investigated different attack algorithms based on paraphrasing [IWG+18], character-level [GLS+18; PDL19b] and word-level [GR20; RDH+19] perturbations, and made these algorithms available in standardized libraries [MLY+20; ZQZ+21].

For the many NLP systems that interact with humans, we argue that *effective adversarial attacks should produce text that is both **valid** and **natural**.* Validity refers to the property that humans perceive the same semantic properties of interest[1] for an adversarial text as for the original text it was produced from. Naturalness refers to the perception that an adversarial text was produced by humans. Adversarial texts that are invalid and/or unnatural can still cause failed NLP model decisions, however, their ultimate effect on humans is negligible because they would fail to convey the intended meaning (e.g. hate speech that is not perceived as hateful) or they would be suspected to be computer-generated (e.g., a phishing email using awkward vocabulary and grammar).

Unfortunately, the scientific literature on adversarial text attacks has neglected (and sometimes ignored) the inclusion of human perception as an essential evaluation criterion – see Table 5.1. We found that (i) 3 studies do not include humans at all in their evaluation; (ii) merely 12 studies consider naturalness, and they only do so under limited settings. Indeed, these studies involve a single attack, one or two naturalness criteria, less than 10 participants, and they disregard the impact of parameters and factors like perturbation size and language proficiency. Instead, the studies rely on automated metrics (i.e cosine distance to measure semantic similarity), but these are not suitable proxies for human perception [MLL+20a].

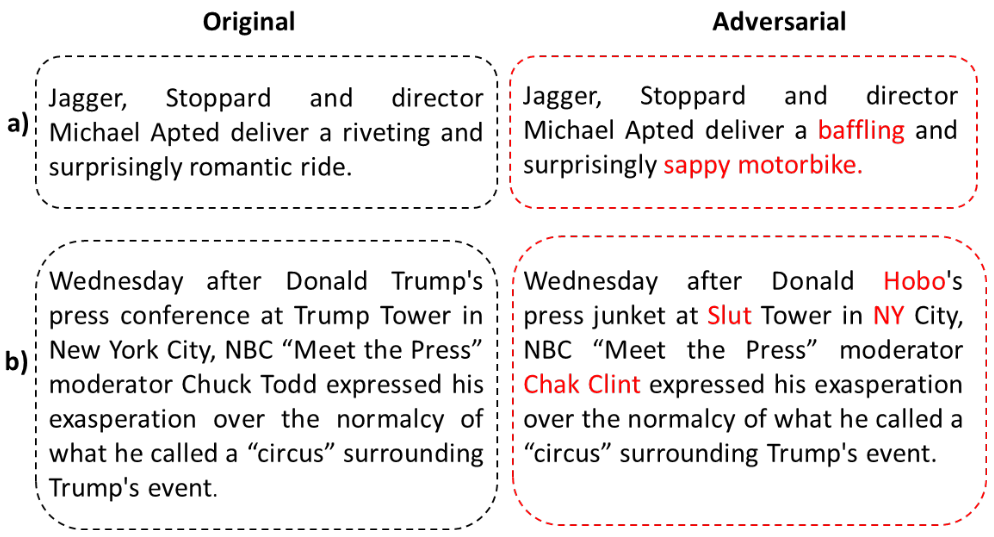
The absence of systematic analysis of adversarial texts *as perceived by humans*

[1]In the case of classification tasks, these semantics properties boil down to the class labels.

risks leading to overestimation of their semantic quality and, in turn, to fallacious model robustness assessment and misguidance during the design of defences. This was hinted in the seminal work from Morris et al. [MLL+20a], where a 10-participant survey on one dataset and two attacks revealed a discrepancy between the human-perceived naturalness of adversarial examples.

Therefore, in this paper, we present the first extensive study that evaluates the human-perceived validity and naturalness of adversarial texts. We surveyed 378 participants in assessing, based on five criteria, over 3000 texts (original and adversarial) coming from three datasets and produced by nine state-of-the-art attacks.

Our investigations first reveal that the participants would classify 28.14% of adversarial examples into a different class than the original example. This means that the adversarial perturbations change human understanding of the modified text and, thus, fail to achieve their purpose. Irrespective of the classification task, participants detect 60.3% of adversarial examples as computer-altered; they can even identify 52.38% of the exact altered word. These findings contrast the overly optimistic conclusions regarding attack success rates from previous small-scale human studies. Our results underline that existing attacks are not effective in real-world scenarios where humans interact with NLP systems. Through our work, we hope to position human perception as a first-class success criterion for text attacks, and provide guidance for research to build effective attack algorithms and, in turn, design appropriate defence mechanisms.



	Original	**Adversarial**
a)	Jagger, Stoppard and director Michael Apted deliver a riveting and surprisingly romantic ride.	Jagger, Stoppard and director Michael Apted deliver a baffling and surprisingly sappy motorbike.
b)	Wednesday after Donald Trump's press conference at Trump Tower in New York City, NBC “Meet the Press” moderator Chuck Todd expressed his exasperation over the normalcy of what he called a “circus” surrounding Trump's event.	Wednesday after Donald Hobo's press junket at Slut Tower in NY City, NBC “Meet the Press” moderator Chak Clint expressed his exasperation over the normalcy of what he called a “circus” surrounding Trump's event.

Figure 5.1: Adversarial examples against NLP model, with perturbations in red. a) Invalid adversarial example generated by Morris et al. [MLL+20a]. b) Unnatural adversarial example generated by Ali et al. [AKA+21].

Table 5.1: Human evaluation performed on quality of adversarial examples by existing literature. The terms abbreviated are Suspiciousness(S.), Detectability(D.), Grammaticality(G.), Meaning(M.). N/A indicates information is not available.

Attack name/paper	**Type**	**Evaluation**					**Participants**	**Attacks studied**
		Validity	**Naturalness**					
			S.	**D.**	**G.**	**M.**		
Hotflip [ERL+18]	Word	✓	X	X	X	X	3	1
Alzantot[ASE+18]		✓	X	X	X	X	20	1
Input-reduction[FWG+18]		✓	X	X	X	X	N/A	1
Kuleshov[KTL+18]		✓	X	X	X	X	5	1
Bae[GR20]		✓	✓	X	✓	X	3	2
Pwws[RDH+19]		✓	✓	X	X	X	6	1
Textfooler [JJZ+20]		✓	X	X	✓	✓	2	1
Bert-attack[LMG+20a]		✓	X	X	✓	X	3	1
Clare [LZP+21]		✓	X	X	X	X	5	2
PSO [ZQY+20]		✓	✓	X	X	X	3	1
Fast-alzantot [JRG+19]		X	X	X	X	X	0	0
IGA [WJH19]		X	X	X	X	X	0	0
Textbugger [LJD+19]	Character	✓	X	✓	X	X	297	1
Pruthi [PDL19b]		✓	X	X	X	X	N/A	1
DeepWordBug [GLS+18]		X	X	X	X	X	0	0
[MLL+20a]	Independent	X	✓	X	✓	✓	10	2
Our study		✓	✓	✓	✓	✓	378	9

5.2 Motivation

Consider the example of fake news shown in Figure 5.1b. (“*Original*”). Ali et al. [AKA+21] have shown that this example is detected by existing fake news detectors based on NLP machine learning models. However, the same authors have also revealed that, if one changes specific words to produce a new sentence (*“Adversarial”*), the same detector would fail to recognize the modified sentence as fake news. This means that fake news could ultimately reach human eyes and propagate.

Fortunately, fake news – like hate speech, spam, phishing, and many other malicious text contents – ultimately targets human eyes and has not only to bypass automated quality gates (such as detectors) but also fool human understanding and judgment. Indeed, to achieve their goal of propagating erroneous information, adversarial fake news should still relay wrong information – they should be “valid” fake news – and be perceived as a text seemingly written by humans – i.e. they should be “natural”. The fake news example from Figure 5.1 is unnatural because it uses irrelevant proper nouns like "Slut Tower" or "Donald Hobo" that do not exist in reality, and this makes the fake news ineffective. We, therefore, argue that invalid and/or unnatural examples do not constitute relevant threats.

Thus, the goal of adversarial text attacks becomes to produce examples that change model decision and are perceived by humans as valid and natural. Our study aims to assess, using human evaluators, whether state-of-the-art text adversarial attacks meet this goal. The answer to this question remains unknown today because, as revealed by our survey of existing attacks (see Table 5.1), only six papers cover both validity and naturalness, five of them do so with less than 10 human participants, and Textbugger [LJD$^+$19] that has the largest number of participants assesses naturalness only at word level, not sentence level. *Nevertheless, all these papers evaluate the effectiveness of the specific attack they introduce (rarely with another baseline) and there is a lack of standardized studies considering them all.*

For our study, the validity and naturalness requirements led us to consider word-based attacks. Indeed, character-based attacks are easily detectable by humans and are even reversible with spelling and grammar check methods [SPV17]. In word-based attacks, the size of the perturbation δ is typically defined as the number of modified words.

5.3 Research questions and metrics

5.3.1 Research questions

Our study first investigates the validity of adversarial examples as perceived by humans.

RQ1 (Validity): Are adversarial examples valid according to human perception?

Validity is the ability of the adversarial example to preserve the class label given to the original text [CGC$^+$22]. Figure 5.1 illustrates a case of an invalid adversarial example, which changes the positive sentiment of the original example. Thus, we aim to compare the label that human participants would give to an adversarial example with the label of the original example. To determine the original label, we use as a reference the "ground truth" label indicated in the original datasets used in our experiments – that is, we assume that this original label is the most likely to be given by human evaluators. To validate this assumption, our study also confronts participants to original examples and checks if they correctly classify these examples (Section 5.5.1). A statistical difference between humans' accuracy on adversarial examples compared to original examples would indicate that a significant portion of adversarial examples is invalid.

In addition to validity, we study next the degree to which adversarial texts are natural.

RQ2 (Naturalness): Are adversarial examples natural?

To answer this question, we measure the ability of humans to suspect that a piece of text has been computer altered (with adversarial perturbations). An adversarial example is thus evaluated as less natural, the more it raises *suspicion* (to have been altered) among the participants.

The suspicion that a text seems computer-altered might arise from different sources, for example the use of specific words, typos, lack of semantic coherence etc. Thus, in addition to evaluating *suspiciousness*, we refine our analysis in order to unveil some reasons why humans may found an adversarial text to be suspicious. We investigate three additional naturalness criteria:

- *Detectability* is the degree to which humans can recognize which words of a given adversarial sentence we altered. High detectability would indicate that the choice of words significantly affect the naturalness of these examples (or lack thereof). We assess detectability in two settings: wherein humans do not know how many words have been altered (unknown $|\delta|$)) and wherein they know the exact number of altered words (known $|\delta|$).
- *Grammaticality* is the degree to which an adversarial text respects the rules of grammar. The presence of grammar errors in a text might raise the suspicion of human evaluators. However, grammar errors may also occur in original (human-written) text. Therefore, we study both the total number of grammar errors in adversarial examples ("error presence"), and the number of introduced errors compared to original texts ("error introduction"). The latter is a better evaluator for the quality of generated adversarial text. A high relative amount of grammar errors could explain the suspiciousness of the adversarial examples (or lack thereof).
- *Meaningfulness* is the degree to which the adversarial text clearly communicates a message that is understandable by the reader. We assess the meaningfulness of adversarial text first in isolation ("clarity")), and then check whether humans believe the meaning of the original text has been preserved under the adversarial perturbation ("preservation"). We hypothesize that adversarial texts with significantly altered meanings are more suspicious.

Because the perturbation size is known to impact success rate and human perceptibility of adversarial attacks in other domains [SDG+21b; DGS+22], we investigate the relationship between the number of altered words and validity/naturalness.

RQ3: How does perturbation size impact the validity and naturalness of adversarial examples?

Although there is a general acceptance that lower perturbation sizes are preferred, the actual magnitude of the effect that perturbation size causes on text perception has not been studied before.

Finally, as we NLP systems interact with a global audience, based on whose language knowledge the adversarial examples might be perceived differently we ask the last research question:

RQ4: How does language proficiency impact the results on validity and naturalness?

So far this aspect is never studied, although we believe it might have an impact on how prone are the user to identify computered altered text.

5.3.2 Reported metrics

Throughout our study, we compute different metrics for each attack separately and all attacks altogether.

1. **Validity:** the percentage of human-assigned labels to adversarial text that match the ground truth provided with the datasets.
2. **Suspiciousness:** the percentage of adversarial texts recognized as "computer altered".
3. **Detectability:** the percentage of perturbed words in an adversarial text that are detected as modified.
4. **Grammaticality:** the percentage of adversarial texts where human evaluators detected present errors (errors introduced by the attack), did not detect or were not sure.
5. **Meaningfulness:** the average value of clarity of meaning and meaning preservation, as measured on a 1-4 Likert scale (the Likert scale options are given in Figure 5.2).

5.3.3 Statistical tests

To assess the significance of differences we observe, we rely on different statistical tests chosen based on the concerned metrics.

- *Proportion tests* are used for validity and suspicion, because they are measured as proportions.
- *Mann Whitney U tests* are used for detectability, grammaticality and meaningfulness because their data are ordinal and may not follow a normal distribution (which this test does not assume). We compute the standardized Z value because our data samples are larger than 30, and the test statistic U is roughly normally distributed.
- *Pearson correlation tests* are used to assess the existence of linear correlations between the perturbation size and validity/naturalness.

We perform all these tests with a significance level of $\alpha = 0.01$.

5.4 Study design

5.4.1 Attacks

To generate the adversarial texts presented to participants, we used the TextAttack library [MLY+20], which is regularly kept up to date with state-of-the-art attacks, including word-based ones.

In total, we used nine word-based attacks from the library. Three of them(*BERTAttack* [LMG+20a], *BAE* [GR20], *CLARE* [LZP+21]) belong to the family of attacks that uses masked language models to introduce perturbations to the original text. Three others (*FGA* [JRG+19], *IGA* [WJH19], *PSO* [ZQY+20]) use evolutionary algorithms to evolve the original text towards an adversarial one. The remaining three (*Kuleshov* [KTL+18], *PWWS* [RDH+19], *TextFooler* [JJZ+20]) use greedy search strategies. For all the attacks, we used the default parameters provided by the original authors. We excluded only Hotflip attack because it was not compatible with the latest Bert-based models and Alzantot attack, for which we used its improved and faster version *FGA*. You can refer to Table 5.1 for details related to the human study performed by the original authors.

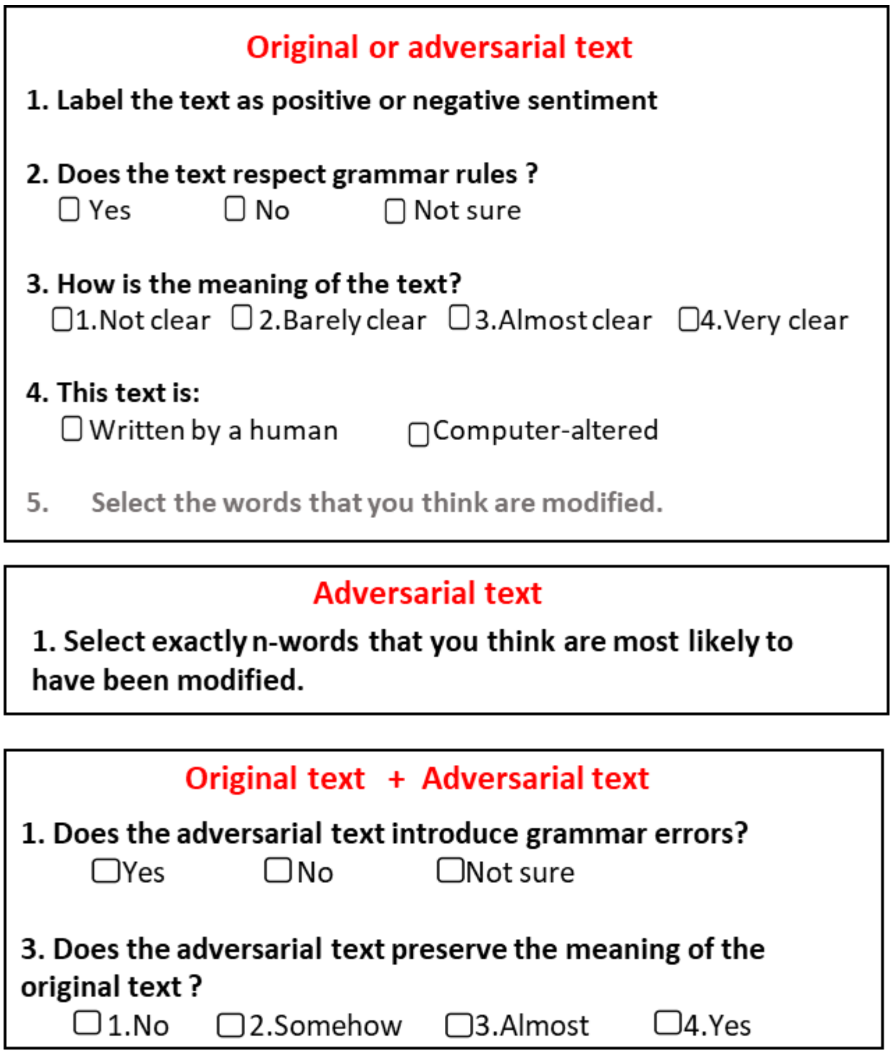
5.4.2 Datasets

We attacked models trained on three sentiment analysis datasets: IMDB movie reviews [MDP+11], Rotten Tomatoes movie reviews [PL05a] and Yelp polarity service reviews [ZZL15a]. We reuse the already available DistilBERT models in the TextAttack library that are trained on these three datasets. Sentiment analysis is a relevant task to assess validity and naturalness, and is easily understandable by any participant, even without domain knowledge. We limited the study to only one task to avoid the extra burden of switching between tasks for the participants. On each dataset, we ran the selected nine word-level attacks, which resulted in 25 283 successful adversarial examples in total.

5.4.3 Questionnaire

We collected the data using an online questionnaire with three parts, presented in Figure 5.2. The beginning of the questionnaire contains the description of computer-altered text as "*a text altered automatically by a program by replacing some words with others*". We do not use the term “adversarial examples” to make the questionnaire accessible to non-technical audiences and avoid biases. We do not provide any hints to participants about the word replacement strategy (i.e. synonym replacement). In addition to this explanation, we clarify to the participants the intended use of the data collected from this study.

The first part of the questionnaire shows examples in isolation and without extra information. It contains questions about validity, suspiciousness, detectability



Original or adversarial text

1. Label the text as positive or negative sentiment

2. Does the text respect grammar rules ?
☐ Yes ☐ No ☐ Not sure

3. How is the meaning of the text?
☐1.Not clear ☐2.Barely clear ☐3.Almost clear ☐4.Very clear

4. This text is:
☐Written by a human ☐Computer-altered

5. Select the words that you think are modified.

Adversarial text

1. Select exactly n-words that you think are most likely to have been modified.

Original text + Adversarial text

1. Does the adversarial text introduce grammar errors?
☐Yes ☐No ☐Not sure

3. Does the adversarial text preserve the meaning of the original text ?
☐1.No ☐2.Somehow ☐3.Almost ☐4.Yes

Figure 5.2: The online questionnaire structure.

(unlimited choices), grammaticality (presence of grammar errors), and meaningfulness (clarity). We display only one text at a time, and each participant receives five random adversarial texts shuffled with five random original texts. We exclude the five original texts used as the initial point for the adversarial generation process, to ensure that participants do not look at two versions of the same text. Question number 5 on detectability will appear only if the participant answers "computer altered" to question 4.

The second part focuses on detectability (exact number). Adversarial examples and their exact number n of perturbed words are shown, and participants have to choose the n words they believe have been altered. Each participant evaluates four adversarial examples they did not see in the first questionnaire part.

The third part shows original and adversarial examples together. It contains questions about grammaticality (errors introduction) and meaning (preservation). Each participant sees the same four adversarial examples (s)he had in the second part and their corresponding original examples.

For each participant, we have (randomly) selected the displayed adversarial examples in order to ensure a balance between the different attacks and perturbation sizes. Each participant sees nine adversarial examples in total (one per attack) with different perturbation sizes (chosen uniformly). More details about this distribution are presented in Appendix 8.2.1.

5.4.4 Participants

In total, 378 adults answered our questionnaire. Among them, 178 were recruited by advertising on private and public communication channels (i.e. LinkedIn, university networks). The rest were recruited through the Prolific crowdsourcing platform. Prolific participants had 80% minimum approval rate and were paid £3 per questionnaire, with an average reward of £9.89/h. All valid Prolific submissions passed two attention checks. For a real-world representation of the population, we advertised the study to targeted English language proficiency levels. As a result, 59 participants had limited working proficiency, 183 had professional proficiency, and 136 were native/bilingual.

You can find the complete dataset with the generated adversarial sentences and the answers from the questionnaire in this link[2].

5.5 Results and Analysis

5.5.1 RQ1: Validity

Validity measurements showed that participants have associated the correct class label (according to the dataset ground truth) to 71.86% of all adversarial examples. This contrasts with original examples, which human participants labeled correctly with 88.78%. This difference is statistically significant (left-tailed proportion test with $Z = -12.79, p = 9.92e - 38$).

Table 5.2 shows the detailed human accuracy numbers for each attack separately. Five of the nine attacks exhibit a statistical difference to original examples (the four others have over 80% of correctly labelled adversarial examples, without significant difference with the original examples). Humans have (almost) the same accuracy as random for two of these attacks, ranging between 50 and 60%.

Insight 1: Five out of nine adversarial attacks generate a significant portion

[2]`https://figshare.com/articles/dataset/ACL_2023_Human_Study_Adversarial_Text_7z/23035472`

Table 5.2: Percentage of correctly labelled adversarial texts as positive or negative sentiment according to the attack method.

Attack	Correctly labelled	Statistical difference with original text
BAE	55.4	X
BERTAttack	71.1	X
CLARE	55.4	X
FGA	84.2	✓
IGA	87.5	✓
Kuleshov	86.8	✓
PSO	63.5	X
PWWS	74.8	X
TextFooler	85.9	✓
All adversarial examples	71.86	✓
Original	88.78	-

(>25%) of adversarial examples that humans would interpret with the wrong label. These examples would not achieve their intended goal in human-checked NLP systems.

5.5.2 RQ2: Naturalness

We report below our results for the different naturalness criteria. The detailed results, globally and for each attack, are shown in Table 5.3.

Suspiciousness

Humans perceive 60.33% of adversarial examples as being computer altered. This is significantly more than the 21.43% of the original examples that raised suspicion (right-tailed proportion test of $Z = 23.63, p = 9.53e^{-124}$). This latter percentage indicates the level of suspiciousness that attacks should target to be considered natural. A per-attack analysis (see Table 5.3) reveals that all attacks produce a significant number of examples perceived unnatural, from 46.55% (FGA) to 68.5% (PSO).

Insight 2: Humans suspect that the majority of the examples (60.33%) produced by adversarial text attacks have been altered by a computer. This demonstrates a lack of naturalness in these examples.

Detectability

When humans are not aware of the perturbation size, they can detect only 45.28% of the altered words in examples they found to be computer altered. This percentage increases to 52.38%, when the actual perturbation size is known (with statistical

Table 5.3: Human evaluation results about the naturalness of adversarial text. Downwards arrows$\downarrow$ indicate lower values are preferred. Upward arrows $\uparrow$ indicate higher values are preferred. Suspicion (S), Detectability(D) and Grammaticality(G) values are percentages, while Meaning(M) values are average of Likert scale items from 1-4.

Attack	**S (%) $\downarrow$**	**D(%)$\downarrow$**		**G(%) $\downarrow$**		**M(1-4) $\uparrow$**	
		Unknown $\lvert\delta\rvert$	Known $\lvert\delta\rvert$	Errors exist	Errors added	Clarity	Preservation
BAE	50.6	35.1	45.3	44.2	29.0	2.64	1.7
BERTAttack	63.9	30.3	44.3	23.7	55.4	2.40	2.07
CLARE	55.9	45.4	39.4	53.8	16.4	2.88	1.7
FGA	46.5	47.5	46.3	44.6	34.5	3.06	2.67
IGA	59.1	53.2	57.8	36.4	47.0	2.70	2.58
Kuleshov	63.9	57.6	65.9	37.6	43.9	2.71	2.09
PSO	68.5	46.7	54.7	37.4	39.1	2.34	1.99
PWWS	65.5	50.3	63.7	34.5	48.0	2.26	2.09
TextFooler	61.5	45.0	54.7	39.1	50.5	2.72	2.47
All examples	60.33	45.28	52.38	38.9	40.6	2.60	2.11

significant according to a Mann-Whitney U Test with $Z = -73.49, p = 4.4e^{-8}$). These conclusions remain valid for all attacks taken individually, with a detection rate ranging from 30.3% to 53.2% (δ unknown) and from 39.4% to 65.9% (δ known).

Insight 3: Humans can detect almost half (45.28%) of the perturbed words in adversarial text. This indicates that the perturbations introduced by attacks are not imperceptible.

Grammaticality

Humans perceive grammar errors in 38.9% of adversarial texts and claim that 40.6% of adversarial texts contain errors not present in their original counterparts. Surprisingly, however, humans are more likely to report grammar errors in examples they perceive as original, than in those they deem computer-altered (73.0% versus 44.6%)(5.4. There is thus no positive correlation between grammaticality and naturalness.

One possible explanation is that human perception of grammar mistakes significantly differs from automated grammar checks. Indeed, the LanguageTool grammar checker [Nab$^+$03] reports that only 17.7% adversarial examples contain errors, which is significantly less than the 40.6% that humans reported. This teaches us that automated grammar checks cannot substitute for human studies to assess grammaticality.

Humans report varying rates of grammar errors across different attacks. The rates are highest for CLARE (53.8%) which is significantly more than the lowest rate (BERTAttack, 23.7%). Human perception of the grammaticality of the different attacks changes drastically when they also see the corresponding original examples (e.g. BERTAttack has the highest error rate with 55.4%, and CLARE has the lowest with 16.4%), indicating again that this criterion is not relevant to explain naturalness.

Please note that the grammar error presence and introduction are studied in two different settings (ref. section 5.3.1 and 5.4.3) with different sets of texts, hence can not be compared against each other. We can only comment on the results separately.

Insight 4: Humans perceive grammar errors in 40% of adversarial examples. However, there is no positive correlation between perceived grammaticality and naturalness.

	Yes	No	Not sure
Computer-altered	44.6	73.0	63.6

Table 5.4: Percentage of adversarial text labelled as computer-altered according to grammar errors

Meaning

Humans give an average rating of 2.60 (on a 1-4 Likert scale) to the meaning clarity of adversarial texts. This is less than original texts, which receives an average rating of 3.44 (with statistical significance based on Mann Whitney U test, with $Z = -412.10, p = 1.43e^{-142}$). Furthermore, participants have mixed opinions regarding meaning preservation from original texts to adversarial texts (average rating of 2.11) on a 1-4 scale.

To check whether lack of clarity indicates a lack of perceived naturalness, we show in Table 5.5, for each rating, the percentage of adversarial texts with this rating that humans perceived as computer altered. We observe a decreasing monotonic relation between rating and suspiciousness. This indicates that the more an adversarial text lacks clarity, the more humans are likely to consider it unnatural.

Table 5.5: Percentage of adversarial texts labelled as computer-altered according to clarity of meaning score

Meaning clarity	1	2	3	4
Computer-altered	86.8	75.7	56.7	25.5

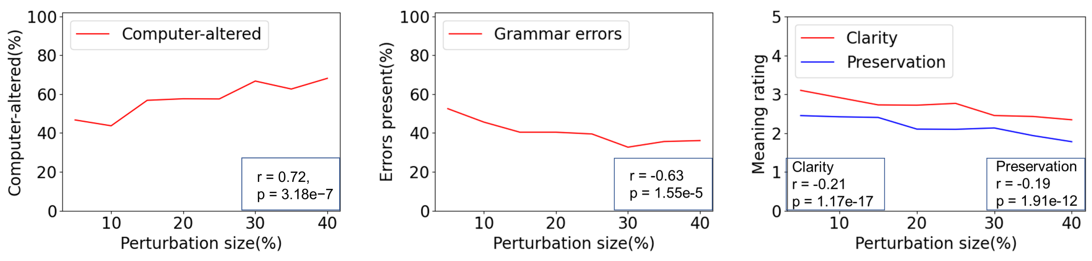
All attacks have an average clarity score ranging from 2.26 (PWWS) to 3.06 (FGA), which tends to confirm the link between naturalness and meaning clarity. Meaning preservation ranges from 1.7 to 2.67. Interestingly, the attacks with a higher preservation rating (FGA, IGA, TextFooler) tends to have a higher validity score (reported in Table5.2), though Kuleshov is an exception.

Insight 5: Humans find adversarial text less clear than original texts, while clarity is an important factor for perceived naturalness. Moreover, attacks that preserve the original meaning tend to produce more valid examples.

5.5.3 RQ3: Impact of perturbation size

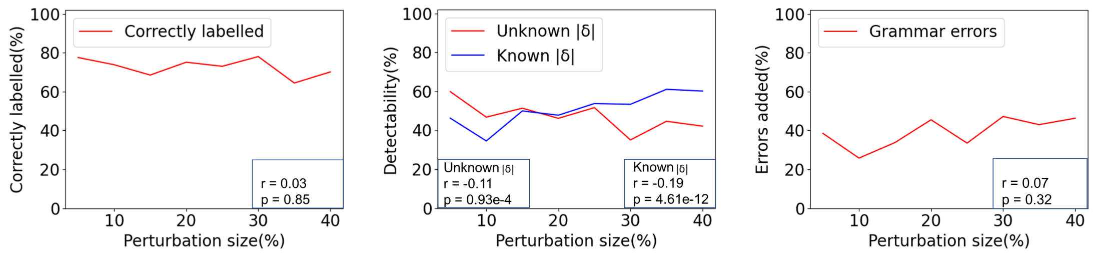
Pearson correlation tests have revealed that perturbation size does not affect validity and detectability (see Figure 5.4), but correlates with suspiciousness, grammaticality and meaning clarity (see Figure 5.3). Thus, adversarial examples are perceived as less natural as more word have been altered (positive correlation). On the contrary, fewer grammatical errors are reported by humans for higher perturbations. We performed an automated check with Language Tool, which gave the opposite results, more grammatical errors are present for larger perturbations. This again demonstrates the mismatch between human perception or knowledge of grammar errors and a predefined set of rules from automatic checkers. However, as a reminder, error presence is not the most relevant metric when evaluating adversarial text. Error introduction should be considered more important. Finally, adversarial examples with larger perturbation size have less clear meaning and preserve less original text's meaning.

Insight 6: The perturbation size negatively affects suspiciousness and meaning, and has no impact on validity or detectability.



(a) Suspiciousness (b) Grammaticality (c) Meaning

Figure 5.3: Effect of perturbation size on Suspiciousness, Grammaticality (Errors present) and Meaning (Clarity and Preservation).



(a) Validity (b) Detectability (c) Grammaticality

Figure 5.4: Effect of perturbation size on Validity, Detectability and Grammaticality (Errors added).

5.5.4 RQ4: Impact of language proficiency

From the results in Table 5.6 we found out that language proficiency only affects some aspects of naturalness and not validity results. People with professional proficiency are more suspicious, they achieve a higher accuracy at detecting adversarial text compared to the other two groups (64.6% vs 54.8% and 57.0%). Regarding grammaticality, people with higher proficiency level report more added errors to the original examples by adversarial attacks. Lastly, for the meaning preservation there is a statistical difference only between two proficiencies, natives give a lower score compared to limited working proficiency.

Insight 7: People with professional proficiency are more suspicious and detect more adversarial texts.

Table 5.6: Effect of language proficiency on the perception of validity and naturalness for human participants. Upward arrows ↑ indicate higher values are preferred. Validity(V), Suspicion(S), Detectability(D) and Grammaticality(G) values are percentages, while Meaning(M) values are average of Likert scale items from 1-4.

Proficiency	**V (%)**	**S (%) ↓**	**D(%)↓**		**G(%) ↓**		**M(1-4) ↑**	
			Unknown $\lvert\delta\rvert$	Known $\lvert\delta\rvert$	Errors exist	Errors added	Clarity	Preservation
Limited	72.4	54.8	42.7	48.6	38.8	31.7	2.65	2.26
Professional	72.6	64.6	43.9	52.5	38.3	40.1	2.63	2.00
Native	70.7	57.0	48.4	53.8	39.7	45.0	2.55	2.15
All examples	71.36	60.33	45.28	52.38	38.9	40.6	2.60	2.11

5.6 Wrap-up and Perspectives

Human validation remains crucial in NLP systems interacting with end users, as automated checks often misalign with human perception. The literature neglects this aspect by focusing more on improving the success rate against the models. We address this issue by conducting the largest human study existing so far, trying to understand the factors affecting human perception of adversarial text.

Our study unveils that a significant portion of adversarial examples produced by state-of-the-art text attacks would not pass human quality gates. These examples are either invalid (labelled differently from intended) or unnatural (perceived as computer altered). This means that the practical success rate of these attacks in systems interacting with humans would be lower than reported in purely model-focused evaluations.

Through our investigations, we confirmed that preserving the meaning of the original text is a key factor for the validity of the adversarial text. *In addition, the factors that raise suspiciousness of a text to have been computer-altered are detectability of (at least one) altered words, as well as meaning clarity.* The (perceived) presence of grammar errors is not a relevant criterion affecting the perceived naturalness of human evaluators. However, grammaticality may still make sense in contexts where exchanged texts rarely contain grammar mistakes (e.g. in professional or formal environments).

More generally, the relevant criteria to evaluate the quality of adversarial examples depend on the considered use case and threat model. We prepared a comprehensive evaluation framework based on individual criteria already existing in the literature, however, they are not necessarily equally important. *Our goal, therefore, is not to qualify an existing attack as "worse than claimed", based on this framework, but rather to raise awareness that different threat scenarios may require different evaluation criteria.* We, therefore, encourage researchers in adversarial attacks to precisely specify which systems and assumptions their study targets, and to justify the choice of evaluation criteria accordingly.

In particular, we corroborate previous studies that discourage the use of automated checks to replace human validation [MLL$^+$20a]. Our study has revealed that human perception of grammaticality does not match the results of grammar-checking tools. *We thus argue that humans play an essential role in the evaluation of adversarial text attacks unless these attacks target specific systems that do not involve or impact humans at all.*

Future work can also address some of the limitations of our study. For example, while we evaluated three datasets and over 3000 sentences, they all targeted the sentiment analysis classification task. Muennighoff et al. [MTM$^+$22] have recently released a large-scale benchmark that covers dozens of text-related tasks and datasets that can further validate our study. It would be especially interesting to consider datasets that use more formal language (i.e. journalistic). Moreover, all our texts and speakers revolve around the English language, while the problems that text adversarial attacks raise (such as fake news and misinformation) are global. Languages with more fluid grammar, that allow more freedom in the positioning of the words, or where subtle changes in tone significantly impact the semantics, can open vulnerabilities and hence require further studies.

Additional studies are required in settings that are not covered by ours including other type of attack strategies, longer sentences, or different distribution for size of perturbation. We focus on word replacement attacks as they are the most common in the literature, however, the human perception of attacks that rely on character insertion or deletion can differ from our conclusions, especially since they are easily confused with typos. For this study we have limited the length of text to 50 words for practical reasons, however the human perception of perturbations in longer texts might differ. And lastly, we considered a uniform distribution of generated adversarial texts per bin for each attack, while their real distribution in the wild might differ from our simulated setting.

Ultimately, we believe that our results shape relevant directions for future research on designing adversarial text. These directions include further understanding the human factors that impact the (im)perceptibility of adversarial examples, and the elaboration of new attacks optimizing these factors (in addition to model failure). The design of relevant attacks constitutes a critical step towards safer NLP models, because understanding systems' security threats paves the way for building appropriate defence mechanisms.

6

Realism of adversarial examples from Deep Generative Models

Deep Generative Models are a potential candidate for generating adversarial examples with low cost for adversarial training. However, despite modeling complex distribution they do not guarantee the satisfaction of domain constraints by their outputs. In this chapter we explore how to inject domain knowledge into these models, such their output space becomes constrained.

Contents

6.1 Introduction

In this chapter, we address the third challenge of the thesis: the lack of general methods to generate constrained adversarial examples in feature space.

Deep Generative Models (DGMs) are powerful tools for generating synthetic tabular data, as they learn to match the distributions of the original data. Their most popular usages are to augment datasets for better predictive performance of ML models ([HWM05]), to remedy data scarcity ([CBM$^+$17]), and to promote fairness ([vBKB$^+$21]), and they are now even used to generate brand-new datasets to ensure privacy in sensitive settings (see, e.g., [JYvdS19; YDvdS20; LHJ$^+$21]).

DGMs have found their way even in the adversarial generation domain by promising shorter generation times compared to iterative attacks, which are the most commonly used nowadays. In this scenario, a target model is incorporated into DGMs' design to transform them into generators of adversarial samples. The outputs of DGMs are sent to the target model to compute adversarial loss, an additional loss term compared to that of standard DGMs. Depending on the specific application, a perturbation loss may also be included to minimize the distance between the generated adversarial example and the original instance. We refer to the DGMs used for the adversarial generation process as AdvDGM. AdvDGMs once trained, can generate adversarial examples in a single pass, making them potentially excellent candidates for model hardening.

Independent of whether DGMs are used in an adversarial setting or not, they need to obey a set of constraints expressing background knowledge about known characteristics of the features and/or existing relationships among them. This is particularly important for tabular data, where such constraints are more explicit. For instance, when synthesizing data from a clinical trial dataset, the value linked to the *"maximum level of hemoglobin recorded"* column must exceed or equal the value associated with the *"minimum level of hemoglobin recorded"* column for each synthetic sample. Indeed, any sample violating such constraint is not realistic as it does not map to real-world medical measurements.

Existing tabular DGMs, can capture complex distributions and generate high-quality data, but they cannot learn these domain constraints independently; hence, they do not guarantee their satisfaction. Constraint satisfaction is, however, a necessary condition for the success of adversarial attacks. The sample rejection strategy for samples that violate constraints, although a possibility, can be impossible for some datasets where the constraint violation rate is 90% to 100% (see. Table 6.1).

In this chapter, we address this limitation by integrating domain knowledge expressed as a set of linear inequalities into standard tabular DGMs by turning them into Constrained Deep Generative Models (C-DGMs). These models guarantee compliant samples with a set of user-defined constraints by incorporating a fully differentiable layer that repairs the violated constraints either at the train or sampling time. We further turn these standard tabular models into Adversarial Constrained Deep Generative Models (AdvCDGMs) which generate only constrained adversarial examples.

To evaluate our approach, we conduct an extensive experimental analysis that involves five tabular DGMs and five tasks for which rich background knowledge is available: our datasets are annotated with up to 31 constraints, each including up to 17 different features. The results empirically demonstrate that we can generate examples with no constraint violation without overhead on the sampling time. Adding the constraint layer also improves the quality of the generated data and the success rate of AdvDGM attacks.

6.2 Research questions and metrics

6.2.1 Research questions

The number of features included in the constraints, together with the type of operations between them, contribute to different levels of difficulty on adhering to these constraints. We hypothesize that current DGMs, can learn some simple relationships between features but fail at complex ones. Therefore, we ask the following research question:

RQ1: To what degree do the current DGMs violate domain constraints?

Our goal in this work is to constrain the output of DGMs by incorporating domain knowledge without harming the quality of the generated data in terms of its utility for training ML models and its indistinguishability from real data by trained classifiers. On the contrary, we hypothesize that the alignment between the generated data distribution and the real distribution through constraint incorporation may improve the quality of the generated data. As a result, we ask the second question:

RQ2: Does the addition of CL layer improve the quality of generated data?

Given the scenario wherein an adversarial example must realistically change the model prediction within a perturbation threshold while adhering to domain constraints, it is clear that these objectives may not always align. Instead, they often can aggressively compete with each other. Therefore, our next research question is:

RQ3: Does the addition of CL layer improve the AdvDGM capability on generating adversarial examples?

One of the premises of using AdvDGMs is that they generate adversarial examples with a single forward pass once trained. This can be very beneficial for hardening models with realistic examples, since most of current attacks follow an iterative process. As such, adding the constraint repair layer should not cause significant overhead and undermine this advantage of AdvCDGMs. We pose the last research question of this chapter:

RQ4: What is the overhead that CL introduces on run time?

This encompasses both the overhead during the training and sampling time.

6.2.2 Metrics

Constraint violation:

To evaluate the performance of standard Deep Generative Models (DGMs) in adhering to predefined constraints, we use the constraints' violation rate (CVR), constraints violation coverage (CVC) and samplewise constraints violation coverage (sCVC).

1. **CVR** represents the percentage of generated synthetic samples in $\mathcal{S}$ that violate the set of constraints Π.

2. **CVC**, represents the percentage of constraints in Π violated by any sample in $\mathcal{S}$.

3. **sCVC** represents the average percentage of constraints violated by each sample in $\mathcal{S}$.

Data quality:

For the usefulness of C-DGMs on generating qualitative synthetic data we use two metrics: utility and detection.

1. **Utility** defines whether our synthetic data can be used as an alternative synthetic dataset to train machine learning models. To evaluate the utility of the generated samples, we follow the *Train on Synthetic, Test on Real* (TSTR) framework [EHR17; JYvdS19]. Following this framework, we train multiple models (e.g., Random Forest, XGBoost etc.), validate them with original training data, and test them on real test data. We report for these models the average F1-score, AUROC, and weighted F1-score.

2. **Detection:** defines whether a classifier can be trained to distinguish the synthetic samples from the real data. We follow the popular evaluation method for tabular data generation (see, e.g., [LQB$^+$22]) and train six models

(e.g., Decision Tree, AdaBoost etc.) to distinguish between the real and synthetic datasets. Again, we report average F1-score, AUROC and weighted F1-score.

Attack performance

To evaluate the attack performance, we use Attack Success Rate (ASR). In addition, we report the clean error rate to compare with the model under no attack.

1. **Clean error rate** refers to the number of original examples labeled incorrectly by the model over the total number of examples.

2. **ASR** refers to the ratio of adversarial examples labeled incorrectly by the model over the total number of examples.

6.3 Problem Statement

6.3.1 Deep Generative Models - DGM

In standard generative modeling, we aim to learn parameters for a generative model (p_θ) that approximates an unknown distribution p_X based on a training dataset $\mathcal{D}$ of N samples drawn from p_X. The DGM model can then output synthetic samples that closely follow the training data distribution. Below, we describe the Generative Adversarial Network (GAN) as a common model architecture in the literature for synthesizing tabular data. Although the focus of this work goes beyond GANs, the example is useful to better understand the problem and the proposed solution.

A GAN consists of two neural networks: a generator G and a discriminator D. The generator takes random noise as input and aims to generate synthetic data samples that are indistinguishable from real data, while the discriminator aims to distinguish between real and fake samples. Both G and D are trained iteratively in a min-max optimization task:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_x}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

where p_z is the noise distribution and V is the function that the discriminator D wants to maximize and generator G wants to minimize.

6.3.2 Adversarial Deep Generative Models - AdvDGM

We introduce a target classifier h for which we want to test the robustness using the examples generated by a DGM. Additionally, let y represent the target class label, and x denote the input sample. The adversarial loss $\mathcal{L}_{adv}$ is computed as:

$$\mathcal{L}_{adv} = \max_{\|\delta\| \leq \epsilon} [\ell(h(p_\theta(x+\delta)), y)]$$

where δ denotes the perturbation added to the input sample x, constrained by its magnitude ϵ such that $|\delta| \leq \epsilon$, and ℓ denotes the loss function used for classification.

Moreover, the perturbation loss $\mathcal{L}_{pert}$ measures the magnitude of modifications required to transform a legitimate sample into an adversarial one. It is calculated as:

$$\mathcal{L}_{pert} = \|\delta\|$$

where δ is the perturbation added to the input sample.

The total loss of the AdvDGM model combines the initial loss of DGMs altogether, with adversarial and perturbation loss as follows:

$$\mathcal{L}_{AdvDGM} = \mathcal{L}_{DGM} - \alpha * \mathcal{L}_{adv} + \beta * \mathcal{L}_{pert}$$

where α and β are scaling factors and $\mathcal{L}_{DGM}$ is specific to the DGM used for modeling the data. For GANs specifically, $\mathcal{L}_{DGM}$ can be defined as:

$$\mathcal{L}_{DGM} = \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)})))$$

6.3.3 Domain constraints

When building datasets, we possess some domain knowledge about the sample space of p_X, including if some features are related to each other. These relationships either existed in the problem space and were propagated to the feature space or were introduced from feature engineering (i.e average operation over several features to create a new feature). Independent of the source, they present some constraints on the values that each feature can take for the generated samples to be acceptable or not. Let Π be a set of constraints expressing this background knowledge. We assume each constraint in Π to be a linear inequality involving variables x_k corresponding to features of the dataset. These inequalities take the form:

$$\sum_k w_k x_k + b \geq 0$$

where w_k are coefficients, b is a constant, and $\geq$ denotes either greater than or equal to. A sample generated by a Deep Generative Model (DGM) assigns values to these variables. If the inequality is true for these assigned values, then the sample satisfies the constraints.

6.3.4 Aim

Our goal is to modify a DGM, into a Constrained Deep Generative Model (C-DGM) to ensure that it generates samples complying with Π and close to those generated by the original DGM. Then, we aim to convert the aforementioned C-DGMs into C-AdvDGMs, which generate constrained adversarial examples to test the robustness of the target classifier h. The objective of the C-AdvDGMs is to minimize L_DGM and the perturbation from the original sample while satisfying the constraints in Π and maximizing the adversarial loss.

6.4 Constrained Deep Generative Models

As mentioned in Chapter 1, the Constrained Layer described in this section was designed and implemented by Mihaela Stoian in [SDC+24]. For completeness purposes, a brief overview of this concept is described below.

The first step to build C-AdvDGMs is building C-DGMs. To ensure that during sample generation our Deep Generative Models (DGMs) comply with predefined user constraints, we introduce a novel differentiable layer called the Constraint Layer (CL). This layer seamlessly integrates into various DGM architectures and guarantees that the generated samples adhere to the given constraints while making minimal alterations to the original outputs of the DGMs.

Consider a scenario where a DGM, such as GAN, produces samples from a noise vector z like in Figure 6.1. These samples, denoted as $\tilde{x}$, often necessitate to be transformed by a mapping function f, into the feature space of the real dataset. An example of such transformation can be an inverse min-max scaling. The transformed sample, $f(\tilde{x})$, undergoes evaluation by our CL, which repairs the violated constraints. The output of CL is the minimally modified $\tilde{x}'$—now compliant with a set of predefined constraints Π. The minimum perturbation is necessary to ensure that the new sample closely follows the original output distribution of the DGM model.

The input required for CL is a variable ordering and a set of constraints expressed as linear inequalities, both provided by the user.

The order in which each feature is processed is defined by the user-defined variable ordering function $\lambda : \mathcal{X} \mapsto [1, D]$, which maps each variable x_i to an integer representing the order of computation. This ordering guides the sequential analysis of features, ensuring that dependencies between variables are properly considered. Such ordering can be random, or defined by some properties of the data, such as casual relationships between them. For simplicity and without loss of generality, we will assume that $\lambda(x_i) = i$.

The CL incrementally adjusts values for each feature x_i on the violated set of constraints, while aiming to minimize changes from the original DGM output. The

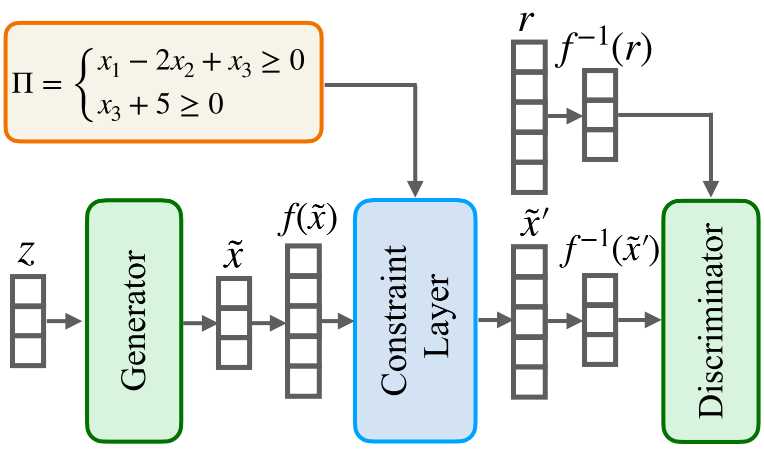


Figure 6.1: Overview on how to integrate CL into a GAN-based model.

decision on what will be the new feature value is based on upper and lower bounds. These bounds represent the maximum and minimum values each feature can take while still satisfying the constraints. They ensure that the generated samples remain within the feasible region defined by the constraints while minimizing changes to the original DGM output. Constraints can impose upper or lower bounds on the value of x_i, or both.

Given λ, the first step of the method (done only once before training) is to compute a set of constraints Π_i associated with each variable x_i. This is done iteratively such that the constraints in Π_i will not contain any variables x_j where $j > i$. To obtain Π_i we either include the existing constraints satisfying this condition above or derive constraints based on simple inequality reductions. The iterative process starts with $Pi_D = Pi$ and continues with computing Π_i, $i < D$, in reverse λ ordering. It ensures that once we fix the values of a feature, we will not risk adjusting it later and breaking any constraints it was initially involved in. Example 6.4.1 illustrates this process.

Example 6.4.1. *In this example, we have a set of constraints* $\Pi = \{x_1 - x_2 \geq 0, x_2 - 5 > 0\}$. *We first obtain* $\Pi_2 = \Pi$ *and, after a simple reduction on the constraints* $x_1 - x_2 \geq 0$ *and* $x_2 - 5 > 0$ *we obtain* $\Pi_1 = \{x_1 - 5 > 0\}$.

Let's denote Π_i^+ as the subset of constraints in Π_i where x_i appears positively, and Π_i^- as the subset where x_i appears negatively. We calculate the least upper bound ub_i as the minimum of the expressions $\varepsilon_i^{\phi}(\mathrm{CL}(\tilde{x}))$, where ε_i^{ϕ} represents the upper bound associated with each constraint $\phi \in \Pi_i^-$. Similarly, for the lower bound lb_i, we compute the greatest lower bound lb_i as the maximum of the expressions $\varepsilon_i^{\phi}(\mathrm{CL}(\tilde{x}))$, where ε_i^{ϕ} represents the lower bound associated with each constraint $\phi \in \Pi_i^+$. We introduce a small positive value, denoted as ϵ, to account for strict inequalities. If a feature x_i violates a strict inequality and its value needs

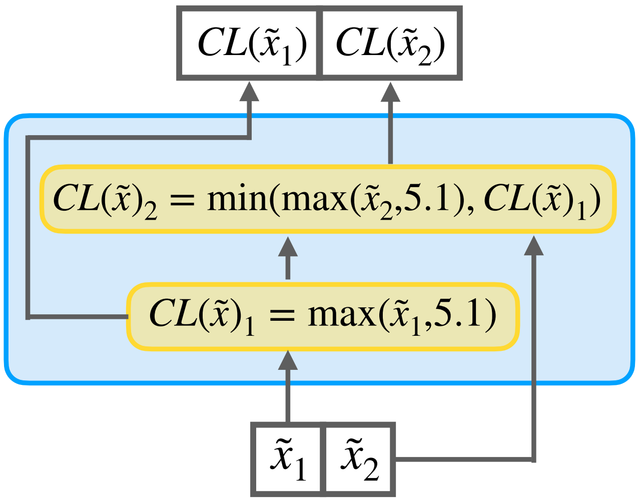

$CL(\tilde{x}_1)$
$CL(\tilde{x}_2)$
$CL(\tilde{x})_2 = \min(\max(\tilde{x}_2, 5.1), CL(\tilde{x})_1)$
$CL(\tilde{x})_1 = \max(\tilde{x}_1, 5.1)$
$\tilde{x}_1$
$\tilde{x}_2$

Figure 6.2: CL in Example 6.4.2.

adjustment, we add ϵ to its lower bound (analogously subtract from the upper bound), ensuring that the constraint is satisfied while keeping the deviation from the original sample minimal.

We can illustrate the inner workings of our constrained layer with Example 6.4.2. A visual representation is given in Figure 6.2.

Example 6.4.2 (Continued). *Given the previous set of constraints,* $\Pi_2 = \Pi$ *and* $\Pi_1 = \{x_1 - 5 > 0\}$. *If* $\tilde{x}_1 = 7$ *and* $\tilde{x}_2 = 3$*, then* $CL(\tilde{x})_1 = 7$ *and* $CL(\tilde{x})_2 = 5.1$*, while if* $\tilde{x}_1 = \tilde{x}_2 = 3$*, then* $CL(\tilde{x})_1 = CL(\tilde{x})_2 = 5.1$*, using* $\epsilon = 0.1$ *to account for strict inequalities.*

Throughout this section, we used GANs as an example of a DGM, however, CL can be integrated with any DGM during training as long as the data transformation function f and its inverse is differentiable. In the case where such a condition is not met or the user deals with a black-box model, the layer can be applied solely at sampling time. In this case, it serves as a postprocessor to guardrail the output of DGMs. We will note such a scenario throughout the text with P-DGM.

6.5 Adversarial Constrained Deep Generative Models

Our C-DGMs serve as the base for building the C-AdvDGMs. In the C-AdvDGM framework, the input to the generator is no longer the noise vector but the initial point x, for which we aim to generate adversarial examples. Continuing with the example of GAN, the generator outputs $\tilde{x}$, which is transformed back into the original data space before undergoing constraint evaluation and repair via the constrained layer, as detailed earlier in 6.4. The resulting constrained example

$\tilde{x}'$ is then transformed into the space used by the GAN before being fed into the discriminator to calculate L_{GAN}. Additionally, $\tilde{x}'$ is transformed by a function g into the space of the target classifier to compute L_{adv}. It's noteworthy that in some cases, $f = g$.

In the literature (see 3.3.2), various choices exist regarding the input and output of the generator. Some papers opt for the original sample alone or concatenated with noise as the starting point, while the output of the generator can either be the perturbation added to the original sample or directly the adversarial sample. Due to the complexity of some of our DGMs, we have chosen to input the original sample and directly output the adversarial one, aiming to streamline the process and minimize runtime. An overview of a transformed C-GAN into C-AdvGAN is given in Figure 6.3

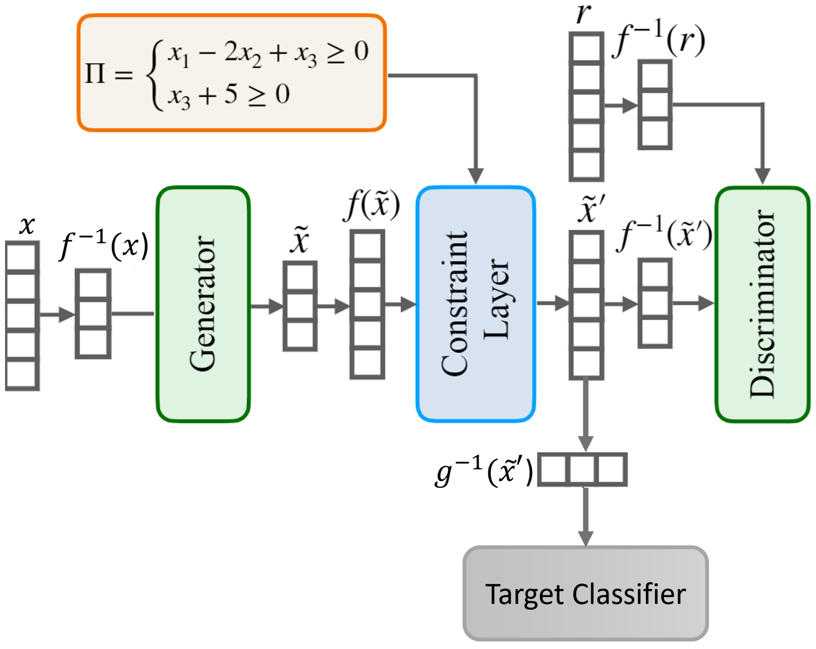


Figure 6.3: Overview of a C-AdvDGM based on GAN.

6.6 Experimental Settings

6.6.1 Models

Our experimentation involved five distinct tabular DGMs: WGAN [ACB17], TableGAN [PMG+18], CTGAN [XSC+19], TVAE [XSC+19], and GOGGLE [LQB+22]. Each of the models comes with its own tabular data transformer which except for GOGGLE and TableGAN were not differentiable. Hence, we first built a differentiable version of the other transformers in PyTorch. As a second step, each model underwent augmentation with our CL, leading to the creation of C-WGAN, C-TableGAN, C-CTGAN, C-TVAE, and C-GOGGLE models. Further, we turn four of these models into C-AdvWGAN, C-AdvTableGAN, C-AdvCTGAN, C-AdvTVAE, and C-AdvGOGGLE. For more details on tabular DGMs architecture, please refer to Appendix 8.3.2.

6.6.2 Datasets

Our evaluation of the models was conducted using real-world datasets, selected based on clear feature relationships either explicitly available or derived from domain expertise. We specifically focused on datasets with a minimum of three known constraints. Consequently, we identified five datasets suitable for evaluation, namely *URL*, *WiDS*, *LCLD*, *Heloc* and *FSP*. These datasets are varied not only in size (ranging from 2K to 1M rows) but also in the number of features (ranging from 24 to 109) and feature types (continuous, categorical, or mixed). The constraints between features exhibit similar diversity in quantity (ranging from 4 to 31) and the number of features involved in each constraint (ranging from 1 to 17). For additional details about the datasets and constraint statistics, please refer to Appendices 8.3.2 and 8.3.2, respectively.

Note: We include the LCLD dataset only in the evaluation of RQ1 and RQ2 in Sections 6.7.1 and 6.7.2. Due to the presence of non-linear constraints that our constrained layer CL cannot yet handle, and for which we cannot guarantee the realism of adversarial examples, we exclude LCLD from the other evaluations in this chapter.

6.6.3 Attack

We use the hyperparameter search method from [SGD+23] and their TorchRLN architecture to build the target model.

For training the adversarial models we performed first a hyperparameter search independently for AdvDGMs and C-DGMs. We explored values of α and β in the range of $[1, 100]$ and learning rate equal to $\{0.001, 0.005, 0.01, 0.05\}$. We used the random variable ordering as an input to the constrained layer CL. Finally, we use an ϵ in $\{0.3, 0.4, 0.5\}$ as the perturbation threshold for the attack.

Note: All the metrics used to evaluate DGMs/AdvCDGMs and their constrained counterparts are reported over 5 runs. For more details on the evaluation procedure for DGMs, refer to Appendix 8.3.2.

6.7 Results

6.7.1 Background Knowledge Alignment

We summarize the results for the Constraint Violation Rate (CVR), Constraints Violation Coverage (CVC) and samplewise Constraints Violation Coverage (sCVC), respectively in Table 6.1, 6.2 and 6.3. The initial five rows represent the results for standard DGMs, while the last two rows belong to their respective constrained versions during training (e.g., C-WGAN, C-TableGAN) and sampling (e.g., P-WGAN, P-TableGAN)

CVR: Our analysis of standard DGMs in Table 6.1 reveals that none of these models guarantee the production of constraint-compliant samples, stressing the necessity for our approach. In over half of the experiments, the CVR exceeds 50%, indicating that more than half of the generated samples violate the constraints. Notably, instances of CVR = 100% are recorded for WGAN when tested on LCLD, and CVR > 95% for four out of five DGMs tested on WiDS. Contrary to standard DGMs, our constrained counterparts C-DGMs consistently yield 0% CVR. This outcome is expected since our constrained DGMs are designed to ensure sample compliance with the constraints.

Table 6.1: Constraint violation rate (CVR) for each model and dataset.

Model/Dataset	URL	WiDS	LCLD	Heloc	FSP
WGAN	11.1±1.6	98.2±0.2	100.0±0.0	57.0±13.0	70.7±8.3
TableGAN	4.9±1.4	96.4±2.4	6.1±0.9	45.6±16.3	71.6±8.7
CTGAN	3.1±2.6	99.9±0.0	11.8±2.7	41.6±12.1	74.3±5.2
TVAE	3.0±0.7	99.9±0.0	3.9±0.5	55.5±1.4	66.4±3.0
GOGGLE	5.9±6.6	78.2±11.6	13.1±2.9	47.3±7.0	63.7±17.6
All P-models	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0
All C-models	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0

CVC: Table 6.2 shows that for 4 out of 5 datasets, all the samples generated by unconstrained DGMs violate at least 50% of the constraints, with CVC reaching up to 100% for WiDS, FSPand News. This entails that the models actually struggle with the majority of the constraints specified, and that the problem cannot be solved by just fixing a very small subset of the available constraints. Meanwhile, all our C-DGMs guarantee that not a single constraint is violated by any of the samples.

sCVC: The results in Table 6.3 show that for all models and datasets, each sample can violate up to 29.8% of the constraints. As a reminder, even a single constraint violated indicates an unfeasible example in a real setting. The unconstrained

Table 6.2: Constraints violation coverage (CVC) for all datasets and models under study.

Model/Dataset	URL	WIDS	LCLD	HELOC	FSP
WGAN	11.1±1.6	100.0±0.0	75.0±0.0	100.0±0.0	75.0±0.0
TableGAN	24.5±3.7	100.0±0.0	50.0±0.0	100.0±0.0	75.0±0.0
CTGAN	16.5±3.8	100.0±0.0	50.0±0.0	99.4±1.3	75.0±0.0
TVAE	12.5±0.0	100.0±0.0	50.0±0.0	100.0±0.0	75.0±0.0
GOGGLE	17.5±6.9	99.2±1.7	50.0±0.0	100.0±0.0	75.0±0.0
All P-models	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0
All C-models	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0

Table 6.3: Samplewise constraints violation coverage (sCVC) for all models and datasets under study.

Model/Dataset	URL	WiDS	Heloc	FSP
WGAN	1.4±0.2	21.2±1.3	29.8±0.2	10.5±2.5
TableGAN	0.6±0.2	14.4±2.8	1.5±0.2	9.0±3.3
CTGAN	0.4±0.3	21.6±0.5	3.0±0.7	7.6±0.3
TVAE	0.4±0.1	21.0±0.2	1.0±0.1	9.4±0.2
GOGGLE	0.8±0.9	10.6±2.4	3.3±0.7	17.2±4.9
All P-models	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0
All C-models	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0	**0.0**±0.0

DGM models learn different representations that produce different rankings for sCVC. For example, WGAN violates on average the most constraints per samples in LCLD. But, for TableGAN, CTGAN, TVAE and GOGGLE, LCLD is among the datasets with the lowest sCVC. Again, all our C-DGMs ensure that sCVC is 0.0, meaning not a single constraint is violated for all the samples.

Visual inspection: Furthermore, we visually contrast the performance of DGMs and C-DGMs by selecting constraints involving two variables and plotting two-dimensional scatter-plots. Figure 6.4 depicts the scatter plot for TableGAN, considering the constraint *MaxHemoglobinLevel* $\geq$ *MinHemoglobinLevel* from the WiDS dataset, with the constraint-violating region highlighted in red. This visualization corroborates our quantitative analysis, showing TableGAN frequently violates the constraint, while C-TableGAN consistently adheres to it. Additionally, we observe that the sample distribution generated by C-TableGAN closely resembles that of the real data.

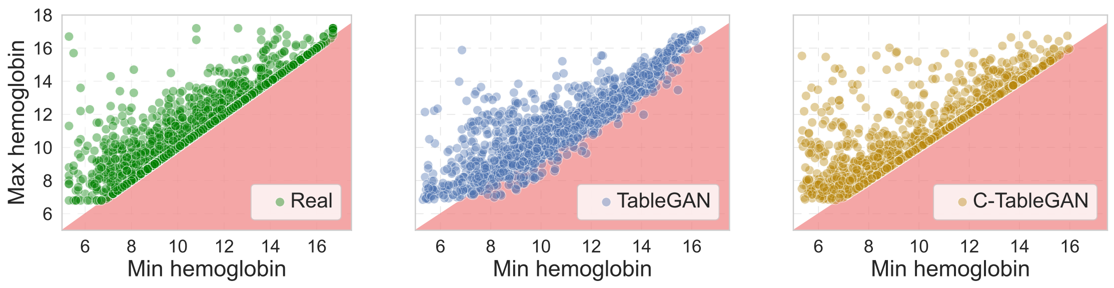


Figure 6.4: Real data and samples generated by TableGAN and C-TableGAN for WiDS.

6.7.2 Synthetic Data Quality

We hypothesized that by including domain knowledge in the sample generation process, the quality of the generated samples will remain intact or improve. To validate our hypothesis, we compare the performance of the standard DGMs with their respective P-DGMs and C-DGMs in terms of utility and detection, as specified in Section 6.6. As a reminder, P-DGM stands for a standard DGM with CL added as a post-processing step at inference time, and C-DGMs stands for a DGM with a CL added at training time. The results of our experiments are summarised in Table 6.4, where we report the average utility performance (left) and the average detection performance (right) over all datasets.

As we can see from Table 6.4, our C-DGMs outperform their standard counterparts and P-DGMs both in terms of utility and detection according to all metrics almost always. Often, such difference in performance is non-negligible, as in the case of GOGGLE where we register a 6.5% difference in terms of utility when calculated in terms of F1-score. The only exception are the three cases: F1 utility of C-CTGAN, together with F1 and *w*F1 of C-TVAE.

On the other hand, the results demonstrate that P-DGMs outperform their standard counterparts 18 times out of 30, thus showing the potential of using CL as a post-processor. The higher performance improvement from C-DGMs compared to P-DGMs is expected as C-DGMs have the advantage of being injected with the background knowledge at training time, thus making them able to exploit it during the generation process.

Note: We report the utility and detection for individual datasets for each model in Appendix 8.3.3.

6.7.3 Adversarial generation capability

In this section, we analyze whether our AdvCDGM models can generate adversarial examples and if the constrained layer helps improve their success rate. Table 6.5 shows the success rate (ASR) of our standard adversarial models and

Table 6.4: Results for every DGM and their constrained versions P-DGM and C-DGM. The best results are in bold.

	Utility (↑)			**Detection (↓)**		
	F1	*w*F1	AUC	F1	*w*F1	AUC
WGAN	0.463	0.730	0.488	0.943	0.952	0.941
P-WGAN	0.462	0.732	0.489	0.926	0.942	0.923
C-WGAN	**0.483**	**0.749**	**0.502**	**0.912**	**0.932**	**0.909**
TableGAN	0.330	0.704	0.400	0.904	0.923	0.902
P-TableGAN	0.328	0.707	0.399	0.897	0.919	0.893
C-TableGAN	**0.375**	**0.714**	**0.432**	**0.891**	**0.911**	**0.887**
CTGAN	**0.517**	0.771	0.532	0.903	0.930	0.908
P-CTGAN	0.512	0.770	0.528	0.897	0.928	0.901
C-CTGAN	0.516	**0.773**	**0.537**	**0.882**	**0.919**	**0.886**
TVAE	0.497	0.767	0.527	**0.868**	**0.894**	0.871
P-TVAE	0.495	0.767	0.525	0.875	0.902	0.876
C-TVAE	**0.507**	**0.773**	**0.537**	0.869	0.901	**0.870**
GOGGLE	0.344	0.624	0.373	0.921	0.936	0.920
P-GOGGLE	0.348	0.626	0.374	**0.919**	0.935	0.918
C-GOGGLE	**0.409**	**0.667**	**0.427**	**0.919**	**0.930**	**0.906**

their constrained counterparts, either at sample time (P-AdvDGM) or train time (C-AdvDGM) for $\epsilon = 0.5$. Please refer to Table 8.12 and 8.12 in Appendix for $\epsilon = 0.4$ and $\epsilon = 0.3$. Despite conducting a hyperparameter search for GOGGLE, we were unable to generate any successful adversarial examples. The model struggled to produce samples below the threshold of $\epsilon = 0.5$. As a result, we exclude it from the analysis below.

The results demonstrate that three out of four unconstrained adversarial models can not increase significantly (if not at all) the error of the target model. Only AdvWGAN and its constrained counterparts manage to do so at high rate for three datasets by achieving an ASR of 0.73 for URL, 0.93 for Heloc and 0.73 for FSP.

Adding the constrained layer as a postprocessing step increases the performance of the AdvDGMs 12 out of 16 times. Although this increase is highly variable in datasets and adversarial models. For example the highest ASR increase is caused by P-AdvWGAN in Heloc with 0.62 points, while for URL no P-AdvDGM model improved the ASR. This can be explained by the fact that according to Table 6.1, all the DGMS had only a small constraint violation rate for URL.

Table 6.5: Attack Success Rate (ASR ↑) of attacks based on tabular Deep Generative Models ($\epsilon = 0.05$)

Attack/Dataset	URL	WiDS	Heloc	FSP
-	0.04	0.19	0.28	0.24
AdvWGAN	**0.73**$\pm$0.10	0.03$\pm$0.00	0.31$\pm$0.16	0.30$\pm$0.19
P-AdvWGAN	**0.73**$\pm$0.10	0.07$\pm$0.08	**0.93**$\pm$0.04	0.70$\pm$0.04
C-AdvWGAN	0.52$\pm$0.16	**0.17**$\pm$0.14	0.46$\pm$0.33	**0.73**$\pm$0.06
AdvTableGAN	**0.14**$\pm$0.08	0.03$\pm$0.00	0.15$\pm$0.04	0.08$\pm$0.03
P-AdvTableGAN	**0.14**$\pm$0.08	**0.17**$\pm$0.01	**0.28**$\pm$0.02	**0.28**$\pm$0.03
C-AdvTableGAN	0.09$\pm$0.01	0.12$\pm$0.02	0.09$\pm$0.19	0.27$\pm$0.01
AdvCTGAN	0.01$\pm$0.00	0.01$\pm$0.01	0.18$\pm$0.03	0.02$\pm$0.03
P-AdvCTGAN	0.01$\pm$0.00	**0.19**$\pm$0.11	0.28$\pm$0.01	0.06$\pm$0.08
C-AdvCTGAN	**0.02**$\pm$0.00	0.16$\pm$0.01	**0.37**$\pm$0.06	**0.32**$\pm$0.02
AdvTVAE	0.00$\pm$0.00	0.00$\pm$0.00	0.18$\pm$0.01	0.06$\pm$0.02
P-AdvTVAE	0.00$\pm$0.00	**0.12**$\pm$0.01	0.32$\pm$0.02	0.23$\pm$0.01
C-AdvTVAE	**0.01**$\pm$0.00	0.10$\pm$0.00	**0.60**$\pm$0.04	**0.28**$\pm$0.01

On the other hand, adding the constrained layer during the training of AdvDGMs increases the performance in 13 out of 16 cases. Although the average improvement brought by C-AdvDGMs of 0.25 is much lower than the 0.57 for P-AdvDGM.

6.7.4 Impact on Run Time

In this section, we investigate the additional computational overhead the constrained layer introduces during the adversarial generation process.

Train time: Table 6.6 presents the average train time across five runs for unconstrained AdvDGMs and their constrained counterparts C-AdvDGMs. For clarity purposes, we have omitted standard deviations, as 81% of them were below 1s and the rest did not exceed 34s.

The results show that the constrained models require at most 4.7 more time to train compared to the unconstrained model (C-AdvWGAN for WiDS). On the other hand, for some models, the constrained and unconstrained versions take the same time to train as in the case of TableGAN. We hypothesize that this is an effect of the data transformers used by each model. TableGAN uses a simple min-max scaling, while the others use one-hot encoding for categorical features and more complex transformations for continuous ones. Every time before and after applying the constrained layer we need to use the data transformer, therefore this causes an overhead. However as in 6.7.3, we saw that in half of the cases, it is better to apply

Table 6.6: Training time in seconds for adversarial models. In brackets is the slow-down factor when using CL

	URL	WiDS	Heloc	FSP
AdvWGAN	135	310	233	70
C-AdvWGAN	319 (2.4)	1472 (4.7)	392 (1.7)	140 (2.0)
AdvTableGAN	64	2843	448	65
C-AdvTableGAN	68 (1.1)	2927 (1.0)	458 (1.0)	67 (1.0)
AdvCTGAN	120	1179	201	47
C-AdvCTGAN	167 (1.4)	1730 (1.5)	413 (2.1)	66 (1.4)
AdvTVAE	63	560	73	24
C-AdvTVAE	122 (1.9)	1240 (2.2)	143 (2.0)	47 (2.0)

Table 6.7: Sample generation time in seconds.

	URL	WiDS	Heloc	FSP
AdvWGAN	0.02	0.10	0.00	0.00
C-AdvWGAN	0.03	0.16	0.00	0.01
AdvTableGAN	0.31	5.51	0.05	0.08
C-AdvTableGAN	0.33	5.63	0.06	0.08
AdvCTGAN	2.09	29.59	0.16	0.26
C-AdvCTGAN	2.08	29.32	0.21	0.26
AdvTVAE	2.08	29.41	0.16	0.26
C-AdvTVAE	2.09	29.20	0.16	0.26

the constrained layer only at sampling time as a postprocessor, which reduces the overhead caused during training.

Sampling time: Table 6.7 presents the average sampling time across five runs for unconstrained AdvDGMs and their constrained counterparts. Again for clarity purposes, we have omitted standard deviations, as 75% of them were 0.00s and the rest did not exceed 0.49s.

From the table results, we observe that C-AdvDGMs exhibit, at most, a 0.12-second increase in runtime compared to their unconstrained counterparts (notably, C-AdvTableGAN for the WiDS dataset). On average, the runtime is 0.02 seconds slower for C-AdvWGAN and 0.04 seconds slower for C-AdvTableGAN. Contrarily, it is 0.06 seconds faster for C-CTGAN and 0.05 seconds faster for C-AdvTVAE. This indicates that our constrained layer incurs negligible overhead, especially when used at sampling time, enabling AdvCDGMs to remain viable for practical applications.

6.8 Wrap-up and Perspectives

In this chapter, we empirically demonstrated that current DGMs often violate domain constraints in the examples they generate. This means their success rate is reduced significantly when used as an adversarial attack method. To address this issue, we introduced a novel layer on top of DGMs that can handle the repair of constraints expressed as linear equalities, thereby yielding Constrained Deep Generative Models (C-DGMs). This allowed us further to build constrained C-AdvDGMs that generate realistic adversarial examples that respect domain constraints.

Our experiments reveal that including our layer during the training or sampling phase improves the success rate of AdvDGMs. This emphasizes the importance of considering compliance with background knowledge as a priority for adversarial attack methods. Although such compliance is not just a means to increase the success rate, rather a strong requirement to properly asses the robustness of ML models.

From the results, we saw that adding the constrained layer during the training or sampling time can be effective in an equal number of cases. However, we lack a clear understanding of why the results are inconsistent and what factors affect them (i.e dataset or constraint properties). *Similar to Chapter 1, we strongly advocate for more visual and analytical methods to better understand the behavior of the adversarial process, their outputs, and their impact on hardening.* These methods would allow us to move beyond superficial empirical results and gain deeper insights into the inner workings of attacks and defenses.

One interesting finding is that among the five models we used as adversarial generators, only WGAN consistently achieved high success rates, both in unconstrained and constrained versions. This is notable because, in the literature on dataset augmentation, WGAN is one of the older models and is not always the most performant. The likely reason for WGAN's success is that uses the same data transformer as the target model: a min-max scaler for continuous data and one-hot encoding (OHE) for categorical data. It remains for the future to test this hypothesis by using a target model with the same scaler as the rest of the AdvDGMs in this work.

Another direction for future work is to expand the set of constraints we can handle beyond linear inequalities. While linear constraints cover a wide range of scenarios, some relationships among features may require more expressive constraint representations. We anticipate significant advancements in this area, where even more complex constraints will be considered.

Although our initial goal was to build C-DGMs that can serve as adversarial generators, our contributions extend beyond realistic adversarial testing for robustness. Constraining the input of the DGM can improve data utility for dataset augmentation, as demonstrated in Section 6.7.2. This is just one of many potential applications for synthetic data. It would be interesting to further evaluate the effect of the constrained layer in other contexts, such as privacy-preserving data generation, rare event simulations, and exploratory analysis. While not all of these applications require realistic outputs by default, we hypothesize that adding domain knowledge through the constrained layer will be beneficial, as seen in the case of utility for dataset augmentation.

7 Conclusion

This chapter presents the overall conclusion of the dissertation and proposes potential research directions for the future.

The main objective of this thesis was to examine the significance of realistic adversarial examples in evaluating and enhancing the robustness of ML systems. Adversarial attacks, a significant security concern, have been the subject of extensive research. However, these studies often overlook the need for adversarial examples to reflect real-world scenarios. This oversight can lead to a skewed understanding of the actual threat level and the risk landscape that models need to defend against.

The Spectrum of Realism in Adversarial Attacks. Adversarial attack techniques vary in their degree of realism. Defining what constitutes a realistic adversarial example is a complex task. We have identified several potential guiding factors in the introduction, such as preserving functionality, ensuring imperceptibility, and altering only a minimal number of reachable features. While we explored human perceptibility for adversarial attacks on NLP in Chapter 5, these concepts often lack universal agreement or understanding and may be approached differently across various domains. For example, L_p norms are commonly used to ensure "imperceptibility" by limiting perturbations and preserving semantics. However, in domains like credit lending systems, even minor changes, such as from "employed" to "unemployed" within a L_p threshold, can drastically alter semantics. Future work could explore deeper into this issue to better understand the spectrum of realism across different domains.

The Role of Domain Knowledge and Constraints. We assumed in Chapter 6 that domain knowledge is articulated through a set of constraints, either included with the datasets or derived by domain experts. While specifying feature types, many datasets often lack comprehensive descriptions of inter-feature relationships, while constraints provided by domain experts may be incomplete. A promising future approach involves using automated methods to extract these relationships. While some existing works can generate constraints as boolean formulas or analytical forms, they often struggle with handling complex constraints. The advancements in large language models and their enhanced mathematical capabilities could offer a more sophisticated approach. A more ambitious vision would be eliminating the need to explicitly incorporate constraints into adversarial generation models, instead allowing the models to learn them implicitly. By observing real data, these models could learn to generate realistic adversarial examples, similar to the role of the discriminator in GANs. However, current GANs primarily focus on learning general distributions rather than the specific realistic properties of the data, as discussed in Chapter 6.

Beyond 'What' Questions. Current studies on adversarial attacks primarily focus on developing new attacks and defenses, leading to a continuous cycle of new defenses being broken by new attacks. Little attention is given to understanding model behaviors against adversarial examples. In Chapter 4, we explored why unrealistic adversarial examples sometimes successfully harden models against realistic ones, suggesting cosine similarity and aggressiveness as potential factors. Yet, these are not exhaustive explanations. In Chapter 6, we observed that adding a layer as post-processing or during training is equally effective. Despite our intuition that constraining during training might lead to local minima, the actual reason remains unknown. There is a need for new quantitative or visual methods to further this understanding. For instance, monitoring the “adversarial landscape” could be as insightful as observing the loss landscape in a normal training process. However, creating loss landscapes is resource-intensive, necessitating efficient adaptation methods for adversarial landscapes.

The Power of ‘Why". If we had more mechanisms to understand attack behaviors during testing and hardening of ML systems, we could leverage these insights to enhance their performance. For instance, if we understood the properties of unrealistic adversarial examples that aid in hardening models against realistic attacks, we could design new attack mechanisms. These mechanisms would guide the attack towards realistic examples (i.e., high cosine similarity and similar degree of aggressiveness), producing examples close in their latent space. This understanding would also inform us when to apply the constraint layer during training or as a post-processor, as discussed in Chapter 6.

This thesis is just a first step to bringing the realism of adversarial examples to the attention of the broad research community on adversarial ML. We hope that other researchers will take up and explore the remaining open directions.

8 Appendices

This chapter includes the appendices related to each individual chapter in the main text.

Contents

8.1 Realism of adversarial attacks for adversarial hardening

8.1.1 Appendix A: Experimental Protocol

Text classification

Datasets. For the evaluation we use three datasets: 1) Rotten Tomatoes movie reviews [PL05b], 2) Offensive subset from TweetEval dataset [BCE+20], 3) AG News [ZZL15b]. For AG News, we use only 7.5k samples for hardening and 1k samples for evaluation. Both Rotten Tomatoes and Tweet Offensive are binary sentiment analysis datasets, while AG News classifies news in four categories. A training sample in these datasets is an input that consists of one or more sentences. The datasets come with different characteristics of input lengths. The smallest dataset is Rotten Tomatoes with 18.49 words in average per input. The largest dataset is AG News with an average of 38.78 words per input.

Model. We use DistilBERT [SDC+19] base model (uncased) available in Hugging Face, because it achieves a comparable performance with BERT, but it is lighter and faster to train. This general pre-trained language model can be fine-tuned in any task. To evaluate the initial robustness of the clean model, we perform normal training for 5 epochs. The model's clean performance is comparable to the state of the art for the dataset used in this study.

Botnet detection

Datasets. For this case study, we use the standard CTU-13 dataset. To avoid training biases, we choose from this dataset bot types with at least two scenarios, one for training and one for testing. This filtering resulted in three bot types, Neris, Rbot and Virut. The selected datasets are feature engineered based on the process from [OSB+19], the same process used by FENCE [CO19]. The datasets are scaled using a Standard Scaler. Neris has 143K training examples (scenario 1,9) and 55K testing examples (scenario 2), with only 0.74% examples labelled in the botnet class. Rbot has 59k training examples (scenario 10,11) and 49k testing examples, with only 0.25% examples labelled as botnet. Virut has 146k training examples and 5k testing examples, with only 0.74% labelled as botnet.

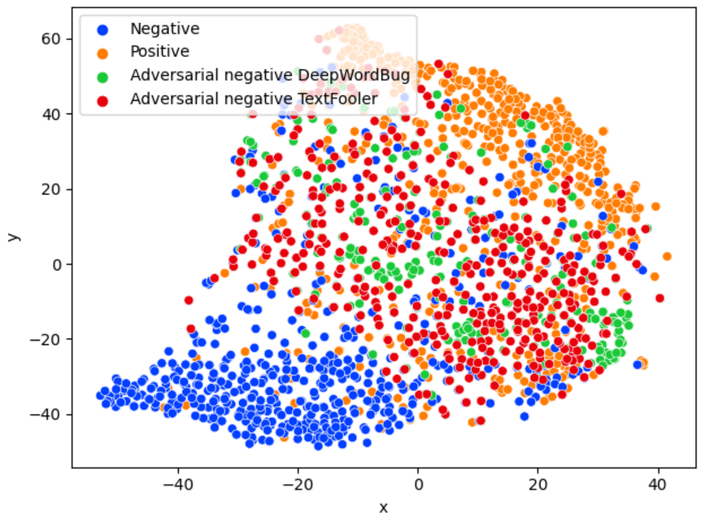
Windows Malware detection

Datasets. We use a collection of benign and malware PE files provided in [AGM+20]. Machine learning classifiers based on static analysis are more prone to detect packing as a sign of maliciousness due to biases in the training set. Aghakhani et al. [AGM+20] showed that if the training set is built such that includes a mix of unpacked benign and packed benign in addition to malicious executables, it is less biased towards detecting specific packing routines as a sign of maliciousness. Therefore, we select in total 4396 packed benign, 4396 unpacked benign, and 8792 malicious samples. We select 70% of files to use for training, 15% for testing, and 15% for adversarial training evaluation. From all our 17584 samples, we extract a set of static features which include: *PE headers*, *PE sections*, *DLL imports*, *API imports*, *Rich Header*, *File generic*. In total there are 24 222 features in this dataset.

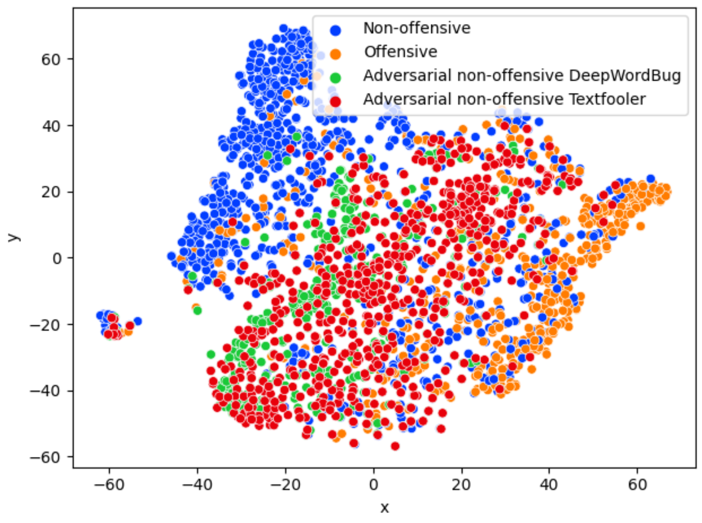
Model. To classify benign and malware software, we used a Random Forest model, a largely used classification algorithm for this task in the literature. We use 100 trees as estimators. This model reached 0.91 AUC-ROC, similar to what other papers reported [AKF+18].

8.1.2 Appendix B: Results

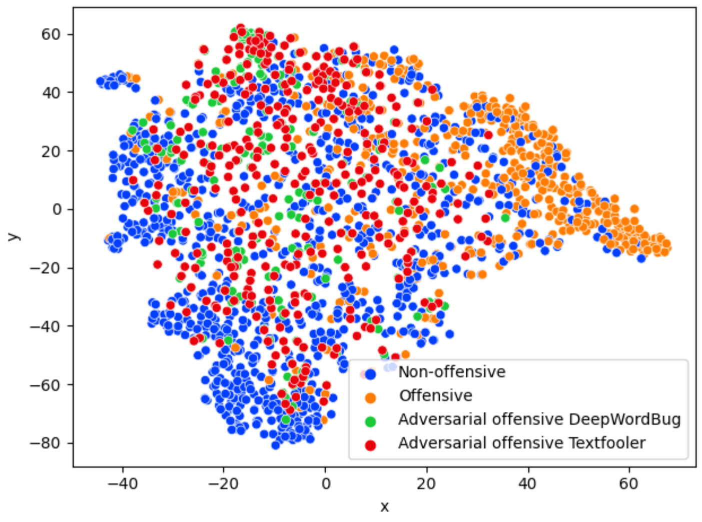
Adversarial embeddings t-SNE plots



(a) Rotten Tomatoes negative reviews



(b) TweetEval (Offensive) non-offensive tweets



(c) TweetEval (Offensive) offensive tweets

Figure 8.1: t-SNE sentence embeddings from the last layer latent representation of the models. The graphs include the original examples and the successful adversarial examples generated by DeepWordBug and TextFooler.

8.2 Realism of adversarial examples in NLP

8.2.1 Distribution of texts to participants in the study

This study was designed to consider the level of perturbation caused by a text. As such, we use the concept of perturbation bins, which are bins of 5% for the perturbation size. As the maximum perturbation we study is 40%, in total there are 8 bins. FGA and IGA attacks set a maximum perturbation size of 20%, therefore we do not consider higher perturbations for them.

Dataset generation: We split the dataset mentioned in Section 5.4.1 in two parts: *original* and *adversarial*, where the original counterpart of adversarial examples in the *adversarial* dataset do not intersect with the original sentences in the *original* dataset. The split is done by randomly selecting first randomly 50 texts for each attack and perturbation bin combination (9x8). In the cases where the attack has generated less than 50 texts in a bin, we take all of those. In total, 3168 texts were added to *adversarial* dataset. To build the *original* dataset, we select from the dataset in Section 5.4.1 the original texts that are not counterparts of the texts in *adversarial* dataset. Finally, the *adversarial* dataset was further split in two parts by selecting randomly the examples.

Survey population: We populate the survey step by step starting from Part 1.

Part 1: Original and Adversarial text
We select 5 original texts randomly from *original* dataset. For adversarial texts, we randomly select 5 attack - perturbation bin combinations from all possible combinations. After that, we choose 5 random texts from these 5 attack-bins from one of two sub-adversarial dataset.

Part 2: We select for each of the 4 attacks not present in Part 1 a random perturbation bin. A random text is then picked for the given attack-bin combination from the sub-dataset of the adversarial dataset that was not picked in Part 1.

Part3: The same adversarial texts as in Part 2, joined with their original counterparts.

The full distribution of the texts to participants is illustrated in Figure 8.2. The distribution of answers per attack and bin is given in Table 8.1 and 8.2 .

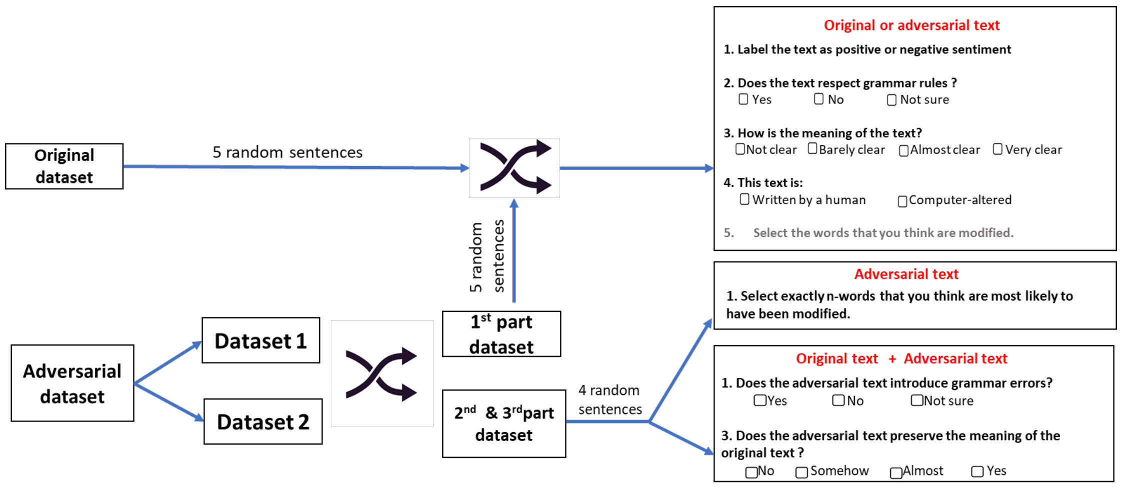

Original dataset
5 random sentences
Adversarial dataset
Dataset 1
Dataset 2
1st part dataset
2nd & 3rd part dataset
5 random sentences
4 random sentences
Original or adversarial text
1. Label the text as positive or negative sentiment
2. Does the text respect grammar rules ?
Yes No Not sure
3. How is the meaning of the text?
Not clear Barely clear Almost clear Very clear
4. This text is:
Written by a human Computer-altered
5. Select the words that you think are modified.
Adversarial text
1. Select exactly n-words that you think are most likely to have been modified.
Original text + Adversarial text
1. Does the adversarial text introduce grammar errors?
Yes No Not sure
3. Does the adversarial text preserve the meaning of the original text ?
No Somehow Almost Yes

Figure 8.2: Distribution procedure of texts to participants of the questionnaire.

Attack \ Bin	5	10	15	20	25	30	35	40
BAE	16	11	19	31	38	29	27	62
BERT	11	8	17	15	22	16	29	76
CLARE	9	13	21	50	30	11	20	41
FGAJia	14	18	22	47	0	0	0	0
IGAWang	13	17	22	36	0	0	0	0
Kuleshov	16	7	10	14	29	27	42	60
PSO	15	10	11	17	23	28	42	76
PWWS	13	12	17	19	30	30	33	72
TextFooler	13	7	7	16	28	27	29	65

Table 8.1: Number of evaluations according to bins for Part 1 of the questionnaire

Attack \ Bin	5	10	15	20	25	30	35	40
BAE	7	8	11	17	33	14	26	29
BERTAttack	12	13	13	21	32	24	29	40
CLARE	15	18	24	55	31	11	16	13
FGA	12	8	12	26	0	0	0	0
IGA	6	12	22	26	0	0	0	0
Kuleshov	12	6	14	29	27	24	32	29
PSO	8	5	14	18	19	31	25	36
PWWSRen	8	7	12	21	22	23	33	26
TextFooler	11	16	17	20	21	32	39	30

Table 8.2: Number of evaluations according to bins for Part 2 and 3 of the questionnaire

8.3 Realism of adversarial examples from Deep Generative Models

8.3.1 Appendix A: Context on collaboration for Chapter 6

To utilize DGMs as an adversarial method for testing the robustness of ML models, while ensuring compliance with background knowledge, we devised a two-step approach:

- Expand DGMs to C-DGMs through a differentiable layer capable of repairing constraints.
- Transform tabular C-DGMs into adversarial generation techniques C-AdvDGMs by performing functional modifications.

However, the first step would require some expertise in Neuro-symbolic AI, which we do not possess. Therefore, we collaborated with researchers from University of Oxford and Vienna University of Technology who have previously worked in this area. Our joint efforts resulted in the publication of our findings as a conference paper [SDC$^+$24]. Some of the results of this work are presented in Section 6.7.1, 6.7.2 and Appendix 8.3.3. The individual contributions to obtain those results are as below:

- Design and implementation of the constrained layer CL (Mihaela Stoian).
- Running experiments for WGAN and GOGGLE models (Mihaela Stoian), for CTGAN, TVAE (Salijona Dyrmishi), for TableGAN (Mihaela Stoian and Salijona Dyrmishi)
- Converting data transforms into differentiable operations for CTGAN & TVAE (Salijona Dyrmishi)

8.3.2 Appendix B: Experimental analysis settings

Models

In our experimental analysis, we use five base models:

- **WGAN** [ACB17] is a GAN model trained with Wasserstein loss in a typical generator discriminator GAN-based architecture. In our implementation, WGAN uses a MinMax transformer for the continuous features and one-hot encoding for categorical ones. It has not been designed specifically for tabular data.
- **TableGAN** [PMG$^+$18] is among the first GAN-based approaches proposed for tabular data generation. In addition to the typical generator and discriminator architecture for GANs, the authors proposed adding a classifier trained to learn the relationship between the labels and the other features. The classifier ensures a higher number of semantically correct produced records. TableGAN uses a MinMax transformer for the features.

- **CTGAN** [XSC$^+$19] uses a conditional generator and training-by-sampling strategy in a generator-discriminator GAN architecture to model tabular data. The conditional generator generates synthetic rows conditioned on one of the discrete columns. The training-by-sampling ensures that the data are sampled according to the log-frequency of each category. Both help to better model the imbalanced categorical columns. CTGAN transforms discrete features using one-hot encoding and a mode-based normalization for continuous features. A variational Gaussian mixture model [CHS18] is used to estimate the number of modes and fit a Gaussian mixture. For each continuous value, a mode is sampled based on probability densities, and its mean and standard deviation are used to normalize the value.
- **TVAE** [XSC$^+$19] was proposed as a variation of the standard Variational AutoEncoder to handle tabular data. It uses the same transformations of data as CTGAN and trains the encoder-decoder architecture using evidence lower-bound (ELBO) loss.
- **GOGGLE**[LQB$^+$22] is a graph-based approach to learning the relational structure of the data as well as functional relationships (dependencies between features). The relational structure of the data is learned by building a graph where nodes are variables and edges indicate dependencies between them. The functional dependencies are learned through a message passing neural network (MPNN). The generative model generates each variable considering its surrounding neighborhood.

Datasets

We use 5 real-world datasets, and an overview of these datasets' statistics can be found in Table 8.3. For the selection, we focused on datasets with at least three feature relationship constraints that either were provided with the description of the datasets or we could derive with our domain expertise. The selected datasets are listed below:

- URL[1] [HY21] is used to perform webpage phishing detection with features describing statistical properties of the URL itself as well as the content of the page.
- WiDS[2] is used to predict if a patient is diagnosed with a particular type of diabetes named Diabetes Mellitus, using data from the first 24 hours of intensive care.
- LCLD[3] is used to predict whether the debt lent is unlikely to be collected. In particular, we use the feature-engineered dataset from [SDG$^+$22a], inspired from the LendingClub loan data. The dataset captures features related to

[1]Link to dataset: https://data.mendeley.com/datasets/c2gw7fy2j4/2
[2]Link to dataset: https://www.kaggle.com/competitions/widsdatathon2021
[3]Link to dataset: https://figshare.com/s/84ae808ce6999fafd192

the loan as well as client history.

- FICO's Home Equity Line of Credit dataset (Heloc[4]) from the FICO xML Challenge is used to predict whether customers will repay their credit lines within 2 years. Similarly to LCLD, the dataset has features related to the credit line and the client's history.
- FSP[5][Bus98] is used to predict 7 types of surface defects in stainless steel plates. The features approximately capture the geometric shape of the defect and its outline.

For URL, WiDS, and LCLD, we used the train-val-test splits provided by [SDG+22a]. For Heloc we used the train-test split of 80-20 and 20% of the training set was later split for validation. Finally, for FSP and News we split the data into 80-10-10% sets, and for the former (which is a multiclass classification dataset) we preserved the class imbalance.

Table 8.3: Datasets statistics.

Dataset	# Train	# Val	# Test	# Feat.	# Cat.	# Cont.	# C.
URL	7K	2K	2K	64	20	44	2
WiDS	22K	6K	7K	109	9	100	2
LCLD	494K	199K	431K	29	8	21	2
Heloc	8K	2K	0.2K	24	8	16	2
FSP	2K	0.2K	0.2K	28	0	28	7

Constraints Datasheet

Here, we outline the structure of our constraints. Let F denote the number of features that appear in at least one constraint, while D represents the total number of features. For any given constraint ϕ, we define F_ϕ^+ as the number of features appearing positively in ϕ, and F_ϕ^- as the number of features appearing negatively. The total number of features appearing in ϕ is $F_\phi = F_\phi^+ + F_\phi^-$.

As illustrated in Table 8.4, the characteristics of constraints vary significantly across different datasets. The number of constraints annotated per dataset ranges from 4 to 31. The percentage of variables appearing in at least one constraint varies from as low as 17.2% in LCLD to as high as 56.88% in WiDS. Additionally, for almost all datasets, the average number of features per constraint is 2.00. However, URL stands out with one constraint involving 17 different features, and LCLD has two constraints where only a single variable appears.

[4]Link to the dataset: https://huggingface.co/datasets/mstz/heloc

[5]Link to dataset: https://www.kaggle.com/datasets/uciml/faulty-steel-plates

Table 8.4: Constraints statistics.

Dataset	# Constr.	F / D	Avg. F_ϕ	Avg. F_ϕ^+	Avg. F_ϕ^-
URL	8	24 / 64	4.25	1.00	3.25
WiDS	31	62 / 109	2.00	1.00	1.00
LCLD	4	5 / 29	1.50	0.75	0.75
Heloc	7	10 / 24	2.00	1.00	1.00
FSP	4	7 / 28	2.00	1.00	1.00

Evaluation protocol for C-DGMs

For evaluating the utility of the DGM/C-DGM models presented in our paper, we followed closely the protocol from [KLP23] which we also reproduce here.

1. First, we generate a synthetic dataset, split into training, validation and test partitions using the same proportions as the real dataset.

2. Then, we perform a hyperparameter search using the synthetic training data partition to train different classifiers/regressors.

   For the binary classification datasets (i.e., URL, WiDS, LCLD, and Heloc) we use: Decision Tree [WKQ+08], AdaBoost [Sch13], Multi-layer Perceptron (MLP) [Hay94], Random Forest [Ho95], XGBoost [CG16], and Logistic Regression [Cox58] classifiers. For multi-class classification datasets, (i.e., FSP) we use Decision Tree, MLP, Random Forest, and XGBoost classifiers. For the regression dataset, (i.e., News) we use MLP, XGBoost, and Random Forest regressors and linear regression. For all the classifiers and regressors above, we considered the same hyperparameter settings as those from Table 26 of [KLP23] and picked the best hyperparameter configuration using the real validation set according to the F1-score.

3. Finally, we tested the selected best models on the real test set and averaged the results across all the classifiers/regressors to get performance measurements for the DGM/C-DGM predictions according to three different metrics: F1-score, weighted F1-score, and Area Under the ROC Curve.

The above procedure was repeated 5 times for each DGM/C-DGM model, and the results were averaged separately for each of the metrics.

For evaluating the DGM/C-DGM models in terms of detection, we slightly adapted the procedure presented by [KLP23]. We first created the training, validation, and test sets by concatenating real and synthetic data, including their targets as usual features, and adding a new target column that specifies whether the data is real or not. By construction, these datasets are binary classification datasets and, thus, are suitable for the hyperparameter search procedure presented in [KLP23] using the 6 different binary classifiers mentioned above. We proceeded to pick the best model using the newly-created validation set, and then we obtained the final detection performance on the newly-created test data (which combines the real and the synthetic data).

8.3.3 Appendix C: Results

Effect on boundary population by CL

To investigate the properties of data generated by C-DGMs, we examine the impact of constraint reparation on the boundary population for the WiDS, HELOC, and FSP datasets, which had the most violated constraints. We defined a band around the boundary with a width w, proportional to the range of the feature values in the real dataset. Since all the considered constraints involve only two features, we set $w = \sqrt{(r_1 p)^2 + (r_2 p)^2}$, where r_1 and r_2 represent the range of the first and second features respectively, and p represents the proportion of the range of values each feature can take. If $p = 1$, the width equals the diagonal of the rectangle defined by the minimum and maximum coordinates.

Table 8.5 shows the percentage of generated samples that lie on the boundary for $p = \{1\%, 5\%, 10\%\}$ for the real dataset (last row), individual DGMs, and C-DGMs, as well as their averages (third and second to last rows, respectively). The results indicate that C-DGMs populate the boundary at a rate much closer to the real data than their unconstrained counterparts. Specifically, the maximum difference between the percentage of real data points on the boundary and the average number of samples generated by C-DGMs is 14.7% (for the WiDS dataset and $p = 1\%$). In contrast, the maximum difference for DGMs is 51.5% (for the Faults dataset and $p = 1\%$).

Full results on DGMs vs. P-DGMs vs C-DGMs

In this section, we present the individual results for each datasets for all the models under study. Respectively for WGAN in Table 8.6, for TableGAN in 8.7, for CTGAN in 8.8, for TVAE in 8.9 and for GOGGLE in 8.10

Real data performance for utility

To ensure meaningful comparisons between our constrained DGM (C-DGM) models and the baseline unconstrained DGMs, we conducted a hyperparameter search. This process allowed us to closely approximate, and sometimes even surpass, the utility performance of real data. We present these results in Table 8.11, using the same evaluation metrics employed for measuring synthetic data utility performance: F1-score, weighted F1-score, and Area Under the ROC Curve. The evaluation followed the protocol outlined in Appendix 8.3.2.

Table 8.5: Percentage of the generated samples that lie on the boundary.

	WiDS			Heloc			FSP		
p	1%	5%	10%	1%	5%	10%	1%	5%	10%
WGAN	28.5	45.2	67.0	22.5	83.9	99.4	14.8	55.6	81.2
C-WGAN	62.1	72.1	83.1	100.0	100.0	100.0	76.8	83.1	89.9
TableGAN	19.7	71.9	88.7	37.4	94.7	99.8	25.3	76.0	95.8
C-TableGAN	72.2	81.1	88.7	83.8	96.1	99.1	75.4	86.2	94.3
CTGAN	6.5	30.5	53.2	80.2	93.1	96.8	2.6	16.4	36.7
C-CTGAN	54.2	62.4	73.8	83.2	96.0	98.3	49.2	62.3	72.9
TVAE	20.2	42.0	62.4	69.4	90.1	96.5	6.5	37.6	56.7
C-TVAE	31.0	56.9	69.7	79.2	92.7	97.1	38.4	57.5	77.4
GOGGLE	0.9	29.1	33.1	46.3	99.6	100.0	35.3	82.2	98.9
C-GOGGLE	7.7	77.2	99.1	83.9	92.4	97.0	81.1	85.4	89.7
DGMs	15.2	43.8	60.9	51.1	92.3	98.5	16.9	53.5	73.9
C-DGMs	45.4	69.9	82.9	86.0	95.4	98.3	64.2	74.9	84.8
Real	60.1	73.3	85.1	85.7	96.5	98.9	68.4	82.8	98.9

Compared to synthetic data, C-WGAN and C-CTGAN models showed utility scores similar to real data. Notably, C-TableGAN and C-TVAE approaches effectively closed the gap between synthetic and real data performance in several instances. For example, in the LCLD dataset, C-TableGAN matched or even slightly exceeded real data performance. However, none of the DGM or C-DGM models approached real FSP data performance. The small size and multiclass nature of the datasets may pose challenges for DGM models in capturing patterns accurately.

AdvDGM attack performance

Table 8.6: Utility and detection results for individual datasets for WGAN.

		Utility (↑)			**Detection (↓)**		
		F1	*w*F1	AUC	F1	*w*F1	AUC
URL	WGAN	0.756	0.764	0.839	0.865	0.856	0.872
	P-WGAN	0.767	0.768	0.840	0.856	0.852	0.867
	C-WGAN	**0.792**	**0.782**	**0.860**	**0.864**	**0.851**	**0.877**
Wids	WGAN	0.329	0.381	0.775	0.975	0.976	0.989
	P-WGAN	0.334	0.386	0.783	0.978	0.978	0.991
	C-WGAN	**0.316**	**0.373**	**0.816**	**0.975**	**0.975**	**0.989**
LCLD	WGAN	**0.239**	**0.359**	**0.618**	**0.999**	**0.999**	**1.000**
	P-WGAN	0.214	0.352	0.617	0.921	0.914	0.942
	C-WGAN	0.232	0.358	0.612	0.923	0.917	0.946
HELOC	WGAN	0.634	0.599	0.677	0.964	0.964	0.983
	P-WGAN	**0.641**	0.600	0.682	0.964	0.965	**0.985**
	C-WGAN	**0.711**	**0.649**	**0.715**	**0.957**	**0.957**	0.975
Faults	WGAN	0.357	0.339	0.742	0.914	0.910	0.916
	P-WGAN	0.355	0.337	0.736	0.910	0.907	0.927
	C-WGAN	**0.367**	**0.349**	**0.742**	**0.843**	**0.845**	**0.874**

Table 8.7: Utility and detection results for individual datasets for TableGAN.

		Utility (↑)			**Detection (↓)**		
		F1	*w*F1	AUC	F1	*w*F1	AUC
URL	TableGAN	0.562	0.659	0.843	0.831	0.822	0.850
	P-TableGAN	0.565	0.660	0.848	0.843	0.823	0.850
	C-TableGAN	**0.611**	**0.695**	**0.868**	**0.849**	**0.835**	**0.856**
Wids	TableGAN	0.171	0.240	0.740	0.963	0.964	0.980
	P-TableGAN	0.173	0.243	0.749	0.962	0.963	0.980
	C-TableGAN	**0.246**	**0.309**	**0.775**	**0.956**	**0.957**	**0.974**
LCLD	TableGAN	**0.123**	**0.286**	**0.587**	**0.895**	**0.895**	**0.926**
	P-TableGAN	0.115	0.281	0.585	0.896	0.888	0.925
	C-TableGAN	0.174	0.314	0.587	0.869	0.871	0.909
HELOC	TableGAN	0.593	0.615	0.707	0.923	0.925	0.953
	P-TableGAN	0.594	0.614	0.707	0.922	0.924	0.953
	C-TableGAN	**0.638**	**0.633**	**0.705**	**0.952**	**0.952**	**0.970**
Faults	TableGAN	0.199	0.199	0.642	0.909	0.906	0.907
	P-TableGAN	0.194	0.195	0.645	0.860	0.868	0.886
	C-TableGAN	**0.208**	**0.209**	**0.635**	**0.830**	**0.817**	**0.849**

Table 8.8: Utility and detection results for individual datasets for CTGAN.

		Utility (↑)			**Detection (↓)**		
		F1	*w*F1	AUC	F1	*w*F1	AUC
URL	CTGAN	0.822	0.799	0.859	0.850	0.862	0.879
	P-CTGAN	0.823	0.802	0.863	0.850	0.861	0.876
	C-CTGAN	**0.830**	**0.816**	**0.877**	**0.820**	**0.839**	**0.864**
Wids	CTGAN	0.362	0.405	0.835	0.990	0.990	0.996
	P-CTGAN	0.365	0.407	0.835	**0.989**	**0.989**	**0.997**
	C-CTGAN	**0.365**	**0.408**	**0.837**	0.990	0.990	0.997
LCLD	CTGAN	**0.290**	**0.407**	**0.657**	0.836	0.847	**0.889**
	P-CTGAN	0.255	0.383	0.652	0.839	0.840	0.892
	C-CTGAN	0.265	0.392	0.641	**0.848**	**0.837**	0.897
HELOC	CTGAN	0.736	0.675	0.744	0.914	0.915	**0.953**
	P-CTGAN	**0.743**	0.678	0.745	**0.913**	**0.914**	0.952
	C-CTGAN	0.736	**0.690**	**0.751**	0.897	0.897	0.943
Faults	CTGAN	0.374	0.372	**0.760**	0.926	0.927	**0.932**
	P-CTGAN	0.372	0.370	0.754	0.895	0.902	0.922
	C-CTGAN	**0.381**	**0.380**	0.759	**0.857**	**0.868**	0.893

Table 8.9: Utility and detection results for individual datasets for TVAE.

		Utility (↑)			**Detection (↓)**		
		F1	*w*F1	AUC	F1	*w*F1	AUC
URL	TVAE	0.810	0.802	0.863	0.813	0.831	0.854
	P-TVAE	0.815	0.806	0.870	**0.806**	**0.829**	**0.850**
	C-TVAE	**0.824**	**0.816**	**0.879**	0.814	0.829	0.849
Wids	TVAE	0.282	0.342	0.800	**0.926**	**0.927**	**0.935**
	P-TVAE	0.285	0.344	0.797	0.964	0.964	0.979
	C-TVAE	**0.322**	**0.378**	**0.815**	0.961	0.962	0.979
LCLD	TVAE	**0.185**	**0.330**	**0.631**	**0.842**	**0.832**	**0.861**
	P-TVAE	0.176	0.325	0.629	0.829	0.825	0.858
	C-TVAE	0.158	0.311	0.633	0.808	0.795	0.840
HELOC	TVAE	**0.735**	**0.696**	**0.752**	**0.914**	**0.916**	**0.947**
	P-TVAE	**0.735**	0.694	0.750	0.909	0.910	0.944
	C-TVAE	0.733	0.690	0.749	0.905	0.908	0.943
Faults	TVAE	0.473	0.463	0.789	**0.843**	**0.847**	**0.872**
	P-TVAE	0.463	0.453	0.788	0.874	0.870	0.895
	C-TVAE	**0.496**	**0.488**	**0.791**	0.855	0.857	0.896

Table 8.10: Utility and detection results for individual datasets for GOGGLE.

		Utility (↑)			**Detection (↓)**		
		F1	*w*F1	AUC	F1	*w*F1	AUC
URL	GOGGLE	0.622	0.648	0.742	0.892	0.880	0.891
	P-GOGGLE	0.626	0.645	0.738	0.884	0.878	0.889
	C-GOGGLE	**0.782**	**0.741**	**0.800**	**0.890**	**0.891**	**0.898**
Wids	GOGGLE	0.189	0.198	0.656	0.987	0.987	0.993
	P-GOGGLE	0.193	0.202	0.665	0.988	0.988	0.994
	C-GOGGLE	**0.185**	**0.253**	**0.675**	**0.972**	**0.971**	**0.984**
LCLD	GOGGLE	**0.163**	**0.315**	**0.543**	0.890	0.892	**0.928**
	P-GOGGLE	0.164	0.315	0.548	0.892	0.889	0.923
	C-GOGGLE	0.219	0.357	0.593	**0.910**	**0.912**	0.943
HELOC	GOGGLE	0.596	0.566	0.600	0.924	0.926	**0.949**
	P-GOGGLE	**0.601**	0.566	0.601	0.926	0.927	0.952
	C-GOGGLE	**0.723**	**0.663**	**0.719**	**0.939**	**0.940**	0.958
Faults	GOGGLE	0.152	0.139	0.577	0.912	0.916	**0.920**
	P-GOGGLE	0.155	0.141	0.578	0.907	0.906	0.917
	C-GOGGLE	**0.136**	**0.121**	**0.546**	**0.884**	**0.817**	0.865

Table 8.11: Utility scores calculated on real data

	F1	*w*F1	AUC
URL	0.884±0.007	0.875±0.014	0.903±0.009
WiDS	0.383±0.021	0.434±0.020	0.832±0.009
LCLD	0.171±0.030	0.316±0.013	0.645±0.007
Heloc	0.772±0.003	0.662±0.011	0.707±0.008
FSP	0.662±0.011	0.659±0.009	0.848±0.010

Table 8.12: Adversarial attack results for $\epsilon = 0.3$

Model	**HELOC**	**FSP**	**URL**	**WIDS**
AdvWGAN	**0.40±0.16**	0.00±0.00	0.27±0.12	0.15±0.10
P-AdvWGAN	**0.40±0.16**	0.00±0.00	**0.75±0.10**	0.34±0.06
C-AdvWGAN	0.27±0.11	**0.02±0.03**	0.39±0.31	**0.43±0.03**
AdvTableGAN	**0.05±0.04**	0.00±0.00	0.15±0.04	0.06±0.03
P-AdvTableGAN	**0.05±0.04**	**0.04±0.02**	**0.28±0.02**	0.17±0.06
C-AdvTableGAN	0.03±0.01	0.02±0.01	0.04±0.09	**0.25±0.02**
AdvCTGAN	0.01±0.00	0.01±0.01	0.18±0.03	0.00±0.01
P-AdvCTGAN	0.01±0.00	**0.08±0.09**	**0.28±0.01**	0.01±0.01
C-AdvCTGAN	**0.02±0.00**	0.01±0.00	0.16±0.05	**0.14±0.04**
AdvTVAE	0.00±0.00	0.00±0.00	0.16±0.01	0.02±0.01
P-AdvTVAE	0.00±0.00	0.00±0.00	0.28±0.02	**0.08±0.01**
C-AdvTVAE	**0.01±0.00**	0.00±0.00	**0.50±0.04**	0.05±0.01

Table 8.13: Adversarial attack results for $\epsilon = 0.4$

Model	**HELOC**	**FSP**	**URL**	**WIDS**
AdvWGAN	**0.59 ± 0.14**	0.02 ± 0.00	0.31 ± 0.15	0.25 ± 0.18
P-AdvWGAN	**0.59 ± 0.14**	**0.13 ± 0.01**	**0.89 ± 0.03**	0.58 ± 0.02
C-AdvWGAN	0.41 ± 0.14	0.11 ± 0.10	0.49 ± 0.34	**0.66 ± 0.06**
AdvTableGAN	**0.09 ± 0.06**	0.02 ± 0.00	0.15 ± 0.04	0.07 ± 0.03
P-AdvTableGAN	**0.09 ± 0.06**	**0.13 ± 0.01**	**0.28 ± 0.02**	**0.26 ± 0.02**
C-AdvTableGAN	0.06 ± 0.01	0.09 ± 0.02	0.06 ± 0.13	**0.26 ± 0.02**
AdvCTGAN	0.01 ± 0.00	0.01 ± 0.01	0.18 ± 0.03	0.01 ± 0.02
P-AdvCTGAN	0.01 ± 0.00	**0.16 ± 0.10**	0.28 ± 0.01	0.03 ± 0.04
C-AdvCTGAN	**0.02 ± 0.00**	0.08 ± 0.01	**0.31 ± 0.06**	**0.24 ± 0.04**
AdvTVAE	0.00 ± 0.00	0.00 ± 0.00	0.18 ± 0.01	0.05 ± 0.02
P-AdvTVAE	0.00 ± 0.00	**0.03 ± 0.00**	0.31 ± 0.02	**0.18 ± 0.01**
C-AdvTVAE	**0.01 ± 0.00**	**0.03 ± 0.00**	**0.62 ± 0.04**	0.15 ± 0.01

List of publications

Papers included in the dissertation

- **Salijona Dyrmishi**, Salah Ghamizi, Thibault Simonetto, Yves Le Traon, and Maxime Cordy. On the empirical effectiveness of unrealistic adversarial hardening against realistic adversarial attacks. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 1384–1400. IEEE, 2023
- **Salijona Dyrmishi**, Salah Ghamizi, and Maxime Cordy. How do humans perceive adversarial text? A reality check on the validity and naturalness of word-based adversarial attacks. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 8822–8836. Association for Computational Linguistics, 2023
- Mihaela Catalina Stoian*, **Salijona Dyrmishi***, Maxime Cordy, Thomas Lukasiewicz, and Eleonora Giunchiglia. How realistic is your synthetic data? constraining deep generative models for tabular data. In *Proceedings of the 12th International Conference on Learning Representations, ICLR 2024, Vienna, Austria, 7–11 May 2024*, 2024

Papers not included in the dissertation

- Thibault Simonetto, **Salijona Dyrmishi**, Salah Ghamizi, Maxime Cordy, and Yves Le Traon. A unified framework for adversarial attack and defense in constrained feature space. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 1313–1319. ijcai.org, 2022

List of figures

List of tables

x

Bibliography

[ACB17] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017. arXiv: `1701.07875` (cited on pages 74, 95).

[ACW18] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018 (cited on page 12).

[AGM^{+}20] Hojjat Aghakhani, Fabio Gritti, Francesco Mecca, Martina Lindorfer, Stefano Ortolani, Davide Balzarotti, Giovanni Vigna, and Christopher Kruegel. When malware is packin'heat; limits of machine learning classifiers based on static analysis features. In *Network and Distributed Systems Security (NDSS) Symposium 2020*, 2020 (cited on pages 39, 90).

[AKA^{+}21] Hassan Ali, Muhammad Suleman Khan, Amer AlGhadhban, Meshari Alazmi, Ahmad Alzamil, Khaled Al-Utaibi, and Junaid Qadir. All your fake detector are belong to us: evaluating adversarial robustness of fake-news detectors under black-box settings. *IEEE Access*, 9:81678–81692, 2021 (cited on pages 49, 50).

[AKF^{+}18] Hyrum S Anderson, Anant Kharkar, Bobby Filar, David Evans, and Phil Roth. Learning to evade static pe machine learning malware models via reinforcement learning. *arXiv preprint arXiv:1801.08917*, 2018 (cited on pages 15, 91).

[ASE^{+}18] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics, 2018. DOI: `10.18653/v1/D18-1316`. URL: `https://aclanthology.org/D18-1316` (cited on page 50).

[BCE+20] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa-Anke, and Leonardo Neves. TweetEval:Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Proceedings of Findings of EMNLP*, 2020 (cited on page 90).

[BCM+13] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013 (cited on page 43).

[Bus98] Massimo Buscema. MetaNet*: the theory of independent judges. *Substance use & misuse*, 33, 1998 (cited on page 97).

[BZZ+21] Tao Bai, Jun Zhao, Jinlin Zhu, Shoudong Han, Jiefeng Chen, Bo Li, and Alex Kot. Ai-gan: attack-inspired generation of adversarial examples. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2543–2547. IEEE, 2021 (cited on page 17).

[CAF+21] Francesco Cartella, Orlando Anunciacao, Yuki Funabiki, Daisuke Yamaguchi, Toru Akishita, and Olivier Elshocht. Adversarial attacks for tabular data: application to fraud detection and imbalanced data. *arXiv preprint arXiv:2101.08030*, 2021 (cited on pages 10, 14, 18).

[CBM+17] Edward Choi, Siddharth Biswal, Bradley A. Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. In *Proceedings of the Machine Learning for Health Care Conference*, 2017 (cited on pages 16, 66).

[CCZ+17] Zhengping Che, Yu Cheng, Shuangfei Zhai, Zhaonan Sun, and Yan Liu. Boosting deep learning risk prediction with generative adversarial networks for electronic health records. In *Proceedings of IEEE International Conference on Data Mining*, 2017 (cited on page 16).

[CG16] Tianqi Chen and Carlos Guestrin. XGBoost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016 (cited on page 98).

[CGC+22] Yangyi Chen, Hongcheng Gao, Ganqu Cui, Fanchao Qi, Longtao Huang, Zhiyuan Liu, and Maosong Sun. Why should adversarial perturbations be imperceptible? rethink the research paradigm in adversarial NLP. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11222–11237, Abu Dhabi, United Arab Emirates. Association for Computational

Linguistics, 2022. URL: https://aclanthology.org/2022.emnlp-main.771 (cited on page 51).

[CHS18] Ramiro Camino, Christian A. Hammerschmidt, and Radu State. Generating multi-categorical samples with generative adversarial networks. *CoRR*, abs/1807.01202, 2018. arXiv: 1807.01202 (cited on page 96).

[CO19] Alesia Chernikova and Alina Oprea. Fence: feasible evasion attacks on neural networks in constrained environments. *arXiv preprint arXiv:1909.10480*, 2019 (cited on pages 14, 15, 18, 20, 25, 32, 33, 90).

[Cox58] David R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20, 1958 (cited on page 98).

[CSD19] Raphael Labaca Castro, Corinna Schmitt, and Gabi Dreo. Aimed: evolving malware with genetic programming to evade detection. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 240–247. IEEE, 2019 (cited on pages 15, 20, 37, 38).

[CW17] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017 (cited on page 14).

[CWC20] Jiyu Chen, David Wang, and Hao Chen. Explore the transformation space for adversarial images. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, pages 109–120, 2020 (cited on page 12).

[DGC23] Salijona Dyrmishi, Salah Ghamizi, and Maxime Cordy. How do humans perceive adversarial text? A reality check on the validity and naturalness of word-based adversarial attacks. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 8822–8836. Association for Computational Linguistics, 2023 (cited on page iii).

[DGS$^+$22] Salijona Dyrmishi, Salah Ghamizi, Thibault Simonetto, Yves Le Traon, and Maxime Cordy. On the empirical effectiveness of unrealistic adversarial hardening against realistic adversarial attacks. *arXiv preprint arXiv:2202.03277*, 2022 (cited on page 52).

[DGS+23] Salijona Dyrmishi, Salah Ghamizi, Thibault Simonetto, Yves Le Traon, and Maxime Cordy. On the empirical effectiveness of unrealistic adversarial hardening against realistic adversarial attacks. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 1384–1400. IEEE, 2023 (cited on page iii).

[DTL21] Yuzhen Ding, Nupur Thakur, and Baoxin Li. Advfoolgen: creating persistent troubles for deep classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 142–151, 2021 (cited on page 17).

[EHR17] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *CoRR*, abs/1706.02633, 2017. arXiv: `1706.02633` (cited on page 68).

[ERL+18] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: white-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics, 2018. DOI: `10.18653/v1/P18-2006`. URL: `https://aclanthology.org/P18-2006` (cited on page 50).

[FWG+18] Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium. Association for Computational Linguistics, 2018. DOI: `10.18653/v1/D18-1407`. URL: `https://aclanthology.org/D18-1407` (cited on page 50).

[GCG+20] Salah Ghamizi, Maxime Cordy, Martin Gubri, Mike Papadakis, Andrey Boystov, Yves Le Traon, and Anne Goujon. Search-based adversarial testing and improvement of constrained credit scoring systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1089–1100, 2020 (cited on pages 3, 14, 15, 18, 25).

[GLS+18] 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 (cited on pages 15, 20, 26, 48, 50).

[GR20] Siddhant Garg and Goutham Ramakrishnan. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics, 2020. DOI: `10.18653/v1/2020.emnlp-main.498`. URL: `https://aclanthology.org/2020.emnlp-main.498` (cited on pages 48, 50, 54).

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014 (cited on pages 11, 14).

[Hay94] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994 (cited on page 98).

[Ho95] Tin Kam Ho. Random decision forests. In *Proceedings of International Conference on Document Analysis and Recognition*, volume 1, 1995 (cited on page 98).

[HT22] Weiwei Hu and Ying Tan. Generating adversarial malware examples for black-box attacks based on gan. In *International Conference on Data Mining and Big Data*, pages 409–423. Springer, 2022 (cited on page 17).

[HWM05] Hui Han, Wenyuan Wang, and Binghuan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *Proceedings of Advances in Intelligent Computing, International Conference on Intelligent Computing*, 2005 (cited on page 66).

[HY21] Abdelhakim Hannousse and Salima Yahiouche. Towards benchmark datasets for machine learning based website phishing detection: an experimental study. *Engineering Applications of Artificial Intelligence*, 104, 2021 (cited on page 96).

[IWG+18] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*, 2018 (cited on page 48).

[JJZ+20] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34 of number 05, pages 8018–8025, 2020 (cited on pages 16, 20, 26, 50, 54).

[JMV+19] Surgan Jandial, Puneet Mangla, Sakshi Varshney, and Vineeth Balasubramanian. Advgan++: harnessing latent layers for adversary generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019 (cited on page 17).

[JRG+19] Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4129–4142, Hong Kong, China. Association for Computational Linguistics, 2019. DOI: `10.18653/v1/D19-1423`. URL: `https://aclanthology.org/D19-1423` (cited on pages 50, 54).

[JSW20] Ahmadreza Jeddi, Mohammad Javad Shafiee, and Alexander Wong. A simple fine-tuning is all you need: towards robust deep learning via adversarial fine-tuning. *arXiv preprint arXiv:2012.13628*, 2020 (cited on page 12).

[JYvdS19] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. PATE-GAN: generating synthetic data with differential privacy guarantees. In *Proceedings of International Conference on Learning Representations*, 2019 (cited on pages 16, 66, 68).

[KBR+23] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling Tabular Data with Diffusion Models. In *Proceedings of International Conference on Machine Learning*, 2023 (cited on page 16).

[KHS+18] Bogdan Kulynych, Jamie Hayes, Nikita Samarin, and Carmela Troncoso. Evading classifiers in discrete domains with provable optimality guarantees. *arXiv preprint arXiv:1810.10939*, 2018 (cited on pages 14, 18).

[KJL+21] Jayoung Kim, Jinsung Jeon, Jaehoon Lee, Jihyeon Hyeong, and Noseong Park. OCT-GAN: Neural ODE-based Conditional Tabular GANs. In *Proceedings of the Web Conference*, 2021 (cited on page 16).

[KLP23] Jayoung Kim, Chaejeong Lee, and Noseong Park. STaSy: Score-based Tabular data Synthesis. In *Proceedings of International Conference on Learning Representations*, 2023 (cited on pages 16, 98, 99).

[KNL20] Bhargav Kuchipudi, Ravi Teja Nannapaneni, and Qi Liao. Adversarial machine learning for spam filters. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–6, 2020 (cited on page 48).

[KTL$^+$18] Volodymyr Kuleshov, Shantanu Thakoor, Tingfung Lau, and Stefano Ermon. Adversarial examples for natural language classification problems, 2018 (cited on pages 50, 54).

[LHJ$^+$21] Jaehoon Lee, Jihyeon Hyeong, Jinsung Jeon, Noseong Park, and Jihoon Cho. Invertible tabular GANs: Killing two birds with one stone for tabular data synthesis. In *Proceedings of Neural Information Processing Systems*, 2021 (cited on pages 16, 66).

[LJD$^+$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 (cited on page 15).

[LJD$^+$19] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: generating adversarial text against real-world applications. In *NETWORK AND DISTRIBUTED SYSTEM SECURITY SYMPOSIUM*, pages 391–406, 2019 (cited on pages 48, 50, 51).

[LLY$^+$20] Jiangnan Li, Jin Young Lee, Yingyuan Yang, Jinyuan Stella Sun, and Kevin Tomsovic. Conaml: constrained adversarial machine learning for cyber-physical systems. *arXiv preprint arXiv:2003.05631*, 2020 (cited on pages 14, 18).

[LLZ$^+$18] Qiang Liu, Pan Li, Wentao Zhao, Wei Cai, Shui Yu, and Victor CM Leung. A survey on security threats and defensive techniques of machine learning: a data driven view. *IEEE access*, 6:12103–12117, 2018 (cited on page 2).

[LMG$^+$20a] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics, 2020. DOI: `10.18653/v1/2020.emnlp-main.500`. URL: `https://aclanthology.org/2020.emnlp-main.500` (cited on pages 48, 50, 54).

[LMG$^+$20b] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. Bert-attack: adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*, 2020 (cited on page 16).

[LMK$^+$20] Eden Levy, Yael Mathov, Ziv Katzir, Asaf Shabtai, and Yuval Elovici. Not all datasets are born equal: on heterogeneous data and adversarial examples. *arXiv preprint arXiv:2010.03180*, 2020 (cited on pages 14, 18).

[LQB+22] Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. GOGGLE: Generative modelling for tabular data by learning relational structure. In *Proceedings of International Conference on Learning Representations*, 2022 (cited on pages 16, 68, 74, 96).

[LSX22] Zilong Lin, Yong Shi, and Zhi Xue. Idsgan: generative adversarial networks for attack generation against intrusion detection. In *Pacific-asia conference on knowledge discovery and data mining*, pages 79–91. Springer, 2022 (cited on page 17).

[LWL+06] Carl Livadas, Robert Walsh, David Lapsley, and W Timothy Strayer. Usilng machine learning technliques to identify botnet traffic. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pages 967–974. IEEE, 2006 (cited on page 32).

[LZP+21] Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. Contextualized perturbation for textual adversarial attack. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5053–5069, Online. Association for Computational Linguistics, 2021. DOI: `10.18653/v1/2021.naacl-main.400`. URL: `https://aclanthology.org/2021.naacl-main.400` (cited on pages 50, 54).

[MDP+11] 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. Association for Computational Linguistics, 2011. URL: `http://www.aclweb.org/anthology/P11-1015` (cited on page 54).

[MLL+20a] John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. Reevaluating adversarial examples in natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3829–3839, Online. Association for Computational Linguistics, 2020. DOI: `10.18653/v1/2020.findings-emnlp.341`. URL: `https://aclanthology.org/2020.findings-emnlp.341` (cited on pages 48–50, 62).

[MLL+20b] John X Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. Reevaluating adversarial examples in natural language. *arXiv preprint arXiv:2004.14174*, 2020 (cited on pages 25, 26).

[MLN+19] Paul Michel, Xian Li, Graham Neubig, and Juan Pino. On evaluation of adversarial perturbations for sequence-to-sequence models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3103–3114, Minneapolis, Minnesota. Association for Computational Linguistics, 2019. DOI: `10.18653/v1/N19-1314`. URL: `https://aclanthology.org/N19-1314` (cited on page 48).

[MLY+20] John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: a framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, 2020 (cited on pages 27, 48, 54).

[MMS+17] 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 (cited on pages 12, 20).

[MTM+22] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: massive text embedding benchmark, 2022. DOI: `10.48550/ARXIV.2210.07316`. URL: `https://arxiv.org/abs/2210.07316` (cited on page 62).

[Nab+03] Daniel Naber et al. A rule-based style and grammar checker, 2003 (cited on page 58).

[NST+18] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, et al. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018 (cited on pages 33, 38).

[OSB+19] Talha Ongun, Timothy Sakharaov, Simona Boboila, Alina Oprea, and Tina Eliassi-Rad. On designing machine learning models for malicious network traffic classification. *arXiv preprint arXiv:1907.04846*, 2019 (cited on page 90).

[PDL19a] Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. Combating adversarial misspellings with robust word recognition. *arXiv preprint arXiv:1905.11268*, 2019 (cited on page 15).

[PDL19b] Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics, 2019. DOI: `10.18653/v1/P19-1561`. URL: `https://aclanthology.org/P19-1561` (cited on pages 48, 50).

[PL05a] Bo Pang and Lillian Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 2005 (cited on page 54).

[PL05b] Bo Pang and Lillian Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*, 2005 (cited on page 90).

[PMG$^+$18] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11, 2018 (cited on pages 16, 74, 95).

[PMJ$^+$16] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016 (cited on page 14).

[PPC$^+$20] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1332–1349. IEEE, 2020 (cited on pages 11, 15, 21, 24, 37, 43).

[QCZ$^+$21] Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics, 2021. DOI: `10.18653/v1/2021.emnlp-main.374`. URL: `https://aclanthology.org/2021.emnlp-main.374` (cited on page 48).

[RDH$^+$19] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097, 2019 (cited on pages 16, 26, 48, 50, 54).

[Sch13] Robert E. Schapire. Explaining AdaBoost. In *Empirical inference.* Springer, 2013 (cited on page 98).

[SDC+19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019 (cited on page 90).

[SDC+24] Mihaela Catalina Stoian*, Salijona Dyrmishi*, Maxime Cordy, Thomas Lukasiewicz, and Eleonora Giunchiglia. How realistic is your synthetic data? constraining deep generative models for tabular data. In *Proceedings of the 12th International Conference on Learning Representations, ICLR 2024, Vienna, Austria, 7–11 May 2024*, 2024 (cited on pages 71, 95, iii).

[SDG+21a] Thibault Simonetto, Salijona Dyrmishi, Salah Ghamizi, Maxime Cordy, and Yves Le Traon. A unified framework for adversarial attack and defense in constrained feature space, 2021. arXiv: `2112.01156 [cs.AI]` (cited on pages 14, 15, 38).

[SDG+21b] Thibault Simonetto, Salijona Dyrmishi, Salah Ghamizi, Maxime Cordy, and Yves Le Traon. A unified framework for adversarial attack and defense in constrained feature space, 2021. DOI: `10.48550/ARXIV.2112.01156`. URL: `https://arxiv.org/abs/2112.01156` (cited on page 52).

[SDG+22a] Thibault Simonetto, Salijona Dyrmishi, Salah Ghamizi, Maxime Cordy, and Yves Le Traon. A unified framework for adversarial attack and defense in constrained feature space. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2022 (cited on pages 96, 97).

[SDG+22b] Thibault Simonetto, Salijona Dyrmishi, Salah Ghamizi, Maxime Cordy, and Yves Le Traon. A unified framework for adversarial attack and defense in constrained feature space. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 1313–1319. ijcai.org, 2022 (cited on page iii).

[SEZ+20] Lea Schönherr, Thorsten Eisenhofer, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. Imperio: robust over-the-air adversarial examples for automatic speech recognition systems. In *Annual Computer Security Applications Conference*, pages 843–855, 2020 (cited on pages 15, 43).

[SGD$^+$23] Thibault Simonetto, Salah Ghamizi, Antoine Desjardins, Maxime Cordy, and Yves Le Traon. Constrained adaptive attacks: realistic evaluation of adversarial examples and robust training of deep neural networks for tabular data. *arXiv preprint arXiv:2311.04503*, 2023 (cited on page 75).

[SHP$^+$21] Ryan Sheatsley, Blaine Hoak, Eric Pauley, Yohan Beugin, Michael J Weisman, and Patrick McDaniel. On the robustness of domain constraints. *arXiv preprint arXiv:2105.08619*, 2021 (cited on pages 14, 15, 18).

[SPV17] Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 366–372, Taipei, Taiwan. Asian Federation of Natural Language Processing, 2017. URL: `https://aclanthology.org/I17-2062` (cited on page 51).

[SPW$^+$20] Ryan Sheatsley, Nicolas Papernot, Michael Weisman, Gunjan Verma, and Patrick McDaniel. Adversarial examples in constrained domains. *arXiv preprint arXiv:2011.01183*, 2020 (cited on pages 14, 18).

[SSK$^+$18] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. *Advances in neural information processing systems*, 31, 2018 (cited on page 17).

[SZS$^+$13] 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 (cited on pages 11, 12).

[TLS$^+$20] Tong Anh Tuan, Hoang Viet Long, Le Hoang Son, Raghvendra Kumar, Ishaani Priyadarshini, and Nguyen Thi Kim Son. Performance evaluation of botnet ddos attack detection using machine learning. *Evolutionary Intelligence*, 13(2):283–294, 2020 (cited on page 32).

[TPS20] Martin Teuffenbach, Ewa Piatkowska, and Paul Smith. Subverting network intrusion detection: crafting adversarial examples accounting for domain-specific constraints. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 301–320. Springer, 2020 (cited on pages 14, 18).

[TWT+20] Yunzhe Tian, Yingdi Wang, Endong Tong, Wenjia Niu, Liang Chang, Qi Alfred Chen, Gang Li, and Jiqiang Liu. Exploring data correlation between feature pairs for generating constraint-based adversarial examples. In *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 430–437. IEEE, 2020 (cited on pages 3, 14, 18).

[UAB19] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni. Survey of machine learning techniques for malware analysis. *Computers & Security*, 81:123–147, 2019 (cited on page 37).

[UAL+19] Muhammad Usama, Muhammad Asim, Siddique Latif, Junaid Qadir, et al. Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems. In *2019 15th international wireless communications & mobile computing conference (IWCMC)*, pages 78–83. IEEE, 2019 (cited on page 17).

[vBKB+21] Boris van Breugel, Trent Kyono, Jeroen Berrevoets, and Mihaela van der Schaar. DECAF: generating fair synthetic data using causally-aware generative networks. In *Proceedings of Neural Information Processing Systems*, 2021 (cited on page 66).

[WJH19] Xiaosen Wang, Hao Jin, and Kun He. Natural language adversarial attacks and defenses in word level. *arXiv preprint arXiv:1909.06723*, 2019 (cited on pages 50, 54).

[WKQ+08] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14, 2008 (cited on page 98).

[XLW+18] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018 (cited on page 16).

[XLZ+18] Chaowei Xiao, Bo Li, Jun-yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3905–3911. International Joint Conferences on Artificial Intelligence Organization, July 2018. DOI: `10.24963/ijcai.2018/543`. URL: `https://doi.org/10.24963/ijcai.2018/543` (cited on page 17).

[XSC+19] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In *Proceedings of Neural Information Processing Systems*, 2019 (cited on pages 16, 74, 96).

[XYW+20] Mingfu Xue, Chengxiang Yuan, Heyi Wu, Yushu Zhang, and Weiqiang Liu. Machine learning security: threats, countermeasures, and evaluations. *IEEE Access*, 8:74720–74742, 2020 (cited on page 2).

[YDvdS20] Jinsung Yoon, Lydia N. Drumright, and Mihaela van der Schaar. Anonymization through data synthesis using generative adversarial networks (ADS-GAN). *IEEE Journal of Biomedical and Health Informatics*, 24, 2020 (cited on page 66).

[ZLW+21] Shuang Zhao, Jing Li, Jianmin Wang, Zhao Zhang, Lin Zhu, and Yong Zhang. Attackgan: adversarial attack against black-box ids using generative adversarial networks. *Procedia Computer Science*, 187:128–133, 2021 (cited on page 17).

[ZQY+20] Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of ACL*, 2020 (cited on pages 50, 54).

[ZQZ+21] Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. Openattack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 363–371, 2021. DOI: `10.18653/v1/2021.acl-demo.43`. URL: `https://aclanthology.org/2021.acl-demo.43` (cited on page 48).

[ZXH+20] Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*, pages 11278–11287. PMLR, 2020 (cited on pages 24, 33).

[ZZL15a] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015 (cited on page 54).

[ZZL15b] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015 (cited on page 90).