A Unified Framework for Adversarial Attack and Defense in Constrained Feature Space

Thibault Simonetto, Salijona Dyrmishi, Salah Ghamizi, Maxime Cordy and Yves Le Traon
University of Luxembourg
{thibault.simonetto, salijona.dyrmishi, salah.ghamizi, maxime.cordy, yves.letraon}@uni.lu

Abstract
The generation of feasible adversarial examples is necessary for properly assessing models that work in constrained feature space. However, it remains a challenging task to enforce constraints into attacks that were designed for computer vision. We propose a unified framework to generate feasible adversarial examples that satisfy given domain constraints. Our framework can handle both linear and non-linear constraints. We instantiate our framework into two algorithms: a gradient-based attack that introduces constraints in the loss function to maximize, and a multi-objective search algorithm that aims for misclassification, perturbation minimization, and constraint satisfaction. We show that our approach is effective in four different domains, with a success rate of up to 100%, where state-of-the-art attacks fail to generate a single feasible example. In addition to adversarial retraining, we propose to introduce engineered non-convex constraints to improve model adversarial robustness. We demonstrate that this new defense is as effective as adversarial retraining. Our framework forms the starting point for research on constrained adversarial attacks and provides relevant baselines and datasets that future research can exploit.

1 Introduction
Research on adversarial examples initially focused on image recognition [Dalvi et al., 2004; Szegedy et al., 2013] but has, since then, demonstrated that the adversarial threat concerns many domains including cybersecurity [Pierazzi et al., 2020; Sheatsley et al., 2020], natural language processing [Alzantot et al., 2018], software security [Yefet et al., 2020], cyber-physical systems [Li et al., 2020], finance [Ghamizi et al., 2020], manufacturing [Mode and Hoque, 2020], and more.

A peculiarity of these domains is that the ML model is integrated in a larger software system that takes as input domain objects (e.g. financial transaction, malware, network traffic). Therefore altering an original example in any direction may result in an example that is infeasible in the real world. This contrasts with images that generally remain valid after slight pixel alterations. Hence, a successful adversarial example should not only fool the model and keep a minimal distance to the original example, but also satisfy the inherent domain constraints.

As a result, generic adversarial attacks that were designed for images – and are unaware of constraints – equally fail to produce feasible adversarial examples in constrained domains [Ghamizi et al., 2020; Tian et al., 2020]. A blind application of these attacks would distort model robustness assessment and prevent the study of proper defense mechanisms.

Problem-space attacks are algorithms that directly manipulate problem objects (e.g. malware code [Aghakhani et al., 2020; Pierazzi et al., 2020], audio files [Du et al., 2020], wireless signal [Sadeghi and Larsson, 2019]) to produce adversarial examples. While these approaches guarantee by construction that they generate feasible examples, they require the specification of domain-specific transformations [Pierazzi et al., 2020]. Their application, therefore, remains confined to the particular domain they were designed for. Additionally, the manipulation and validation of problem objects are computationally more complex than working with feature vectors.

An alternative to problem-space attacks is feature-space attacks that enforce the satisfaction of the domain constraints. Some approaches for constrained feature space attacks modify generic gradient-based attacks to account for constraints [Sheatsley et al., 2020; Tian et al., 2020; Erdemir et al., 2021] but are limited to a strict subset of the constraints that occurs in real-world applications (read more in Appendix A of our extended version\(^1\), where we discuss the related work thoroughly). Other approaches tailored to a specific domain manage to produce feasible examples [Chernikova and Oprea, 2019; Li et al., 2020; Ghamizi et al., 2020] but would require drastic modifications throughout all their components to be transferred to other domains. To this date, there is a lack of generic attacks for robustness assessment of domain-specific models and a lack of cross-domain evaluation of defense mechanisms.

In this paper, we propose a unified framework\(^2\) for constrained feature-space attacks that applies to different domains without tuning and ensures the production of feasible examples. Based on our review of the literature and our analysis of the covered application domains, we propose a

---

\(^1\)Extended version: https://arxiv.org/abs/2112.01156.
\(^2\)https://github.com/serval-uni-lu/moeva2-ijcai22-replication
Let us consider a classification problem defined over an input space $Z$. Each input $z \in Z$ is an object of the considered application domain (e.g. malware [Aghakhani et al., 2020], network data [Chernikova and Oprea, 2019], financial transactions [Ghamizi et al., 2020]). We assume the existence of a feature mapping function $\varphi : Z \rightarrow R^n$ that maps $Z$ to an $n$-dimensional feature space $X$ over the feature set $F = \{f_1, f_2, \ldots, f_n\}$. For simplicity, we assume $X$ to be normalized such that $X \subseteq \{0,1\}^n$. That is, for all $z \in Z$, $\varphi(z)$ is an m-sized feature vector $x = (x_1, \ldots, x_m)$ where $x_i \in \{0,1\}$ and is the j-th feature. Each object $z$ respects some natural constraints in order to be valid. In the feature space, these conditions translate into a set of constraints over the feature values, which we denote by $\Omega$. By construction, any feature vector $x$ generated from a real-world object $z$ satisfies all constraints $\omega \in \Omega$.

Based on our review of the literature, we have designed a constraint language to capture and generalize the types of feature constraints that occur in the surveyed domains. Our framework allows the definition of constraint formulae according to the following grammar:

$$\omega := \omega_1 \land \omega_2 \mid \omega_1 \lor \omega_2 \mid \psi_1 \geq \psi_2 \mid f \in \{\psi_1, \ldots, \psi_k\}$$

$$\psi := c \mid f \mid \psi_1 \oplus \psi_2 \mid x_i$$

where $f \in F$, $c$ is a constant real value, $\omega, \omega_1, \omega_2$ are constraint formulae, $\geq \in \{<, \leq, =, \neq, \geq, >\}$, $\psi, \psi_1, \ldots, \psi_k$ are numeric expressions, $\oplus \in \{+, -, *, /\}$, and $x_i$ is the value of the i-th feature of the original input $x$.

One can observe from the above that our formalism captures, in particular, feature boundaries (e.g. $f > 0$) and numerical relationships between features (e.g. $f_1/f_2 < f_3$) – two forms of constraints that have been extensively used in the literature [Chernikova and Oprea, 2019; Ghamizi et al., 2020; Tian et al., 2020; Li et al., 2020].

2.2 Threat Model and Attack Objective

Let a function $H : X \rightarrow Y$ be a binary classifier and function $h : X \rightarrow R$ be a single output predictor that predicts a continuous probability score. Given a classification threshold $t$, we can induce $H$ from $h$ with, $H(x) = I_h(x) > T$, where $I_h$ is an indicator function, that is, $I$ outputs 1 if the probability score is equal or above the threshold and 0 otherwise.

In our threat model, we assume that the attacker has knowledge of $h$ and its parameters, as well as of $F$ and $\Omega$. We also assume that the attacker can directly modify a subset of the feature vector $x = (x_1 \ldots x_m)$, with $m < n$. We refer to this subset as the set of mutable features. The attacker can only feed the example to the system if this example satisfies $\Omega$.

Given an original example $x$, the attack objective is to generate an adversarial example $x + \delta$ such that $H(x + \delta) \neq H(x)$, $\delta < \epsilon$ for a maximal perturbation threshold $\epsilon$ under a given $p$-norm, and $x + \delta \in X$. While domain constraints guarantee that an example is feasible, (e.g. total credit amount must be equal to the monthly payment times the duration in months), we limit the maximum perturbation to produce imperceptible adversarial examples. By convention, one may prefer $p = \infty$ for continuous features, $p = 1$ for binary features and $p = 2$ for a combination of continuous and binary features. We refer to such examples $x + \delta$ as a constrained adversarial example. We also name constrained adversarial attack algorithms that aim to produce the above optimal constrained adversarial example. We propose two such attacks.
However, as our experiments reveal (see Table 2 and Section 6), a straight application of PGD does not manage to generate any example that satisfies $\Omega$. This raises the need to equip PGD with the means of handling domain constraints.

An out-of-the-box solution that we have experimented is to pair PGD with a mathematical programming solver, i.e. Gurobi [Gurobi Optimization, LLC, 2022]. Once PGD managed to generate an adversarial example (not satisfying the constraint), we invoke the solver to find a solution to the set of constraints close to the example that PGD generated (and under a perturbation sphere of $\epsilon$ size). Unfortunately, this solution does not work out either because the updated examples do not fool the classifier anymore or the solver simply cannot find an optimal solution given the perturbation size.

In face of this failure, we conclude that this gradient-based attack cannot generate constrained adversarial examples if we do not revisit its fundamentals in light of the new attack objective. We, therefore, propose to develop a new method that considers the satisfaction of constraints as an integral part of the perturbation computation.

Concretely, we define a penalty function that represents how far an example $x$ is from satisfying the constraints. More precisely, we express each constraint $\omega_i$ as a penalty function $\text{penalty}(x, \omega_i)$ over $x$ such that $x$ satisfies $\omega_i$ if and only if $\text{penalty}(x, \omega_i) \leq 0$. Table 1 shows how each constraint formula to penalty functions. $\tau$ is an infinitesimal value.

### Table 1: From constraint formulae to penalty functions

| Constraints formulae | Penalty function |
|----------------------|------------------|
| $\omega_1 \land \omega_2$ | $\omega_1 + \omega_2$ |
| $\omega_1 \lor \omega_2$ | $\min(\omega_1, \omega_2)$ |
| $\psi \in \Psi = \{\psi_1, \ldots, \psi_k\}$ | $\min\{\psi_i \in \Psi : |\psi - \psi_i|\}$ |
| $\psi_1 \leq \psi_2$ | $\max(0, \psi_1 - \psi_2)$ |
| $\psi_1 < \psi_2$ | $\max(0, \psi_1 - \psi_2 + \tau)$ |
| $\psi_1 = \psi_2$ | $|\psi_1 - \psi_2|$ |

With $x^0 = x$, and $R$ a repair function. At each iteration $t$, $R$ updates the features of the example to repair the non-convex constraints whose penalty functions are not back-propagated with the gradient $\nabla_x l$ (if any).

## 4 Multi-Objective Generation of Constrained Adversarial Examples

As an alternative to C-PGD, we propose MoEvA2, a multi-objective optimization algorithm whose fitness function is driven by the attack objective described in Section 2.

### 4.1 Objective Function

We express the generation of constrained adversarial examples as a multi-objective optimization problem that reflects three requirements: misclassification of the example, maximal distance to the original example, and satisfaction of the domain constraints. By convention, we express these three objectives as a minimization problem.

The first objective of a constrained attack is to cause misclassification by the model. When provided an input $x$, the binary classifier $H$ outputs $h(x)$, the prediction probability that $x$ lies in class 1. If $h(x)$ is above the classification threshold $t$, the model classifies $x$ as 1; otherwise as 0. Without knowledge of $t$, we consider $h(x)$ to be the distance of $x$ to class 0. By minimising $h(x)$, we increase the likelihood that the $H$ misclassifies the example irrespective of $t$. Hence, the first objective that MoEvA2 minimizes is $g_1(x) \equiv h(x)$.

The second objective is to minimize perturbation between the original example $x^0$ and the adversarial example, to limit the perceptibility of the crafted perturbations. We use the conventional $L_p$ distance to measure this perturbation. The second objective is $g_2(x) \equiv L_p(x - x^0)$.

The third objective is to satisfy the domain constraints. Here, we reuse the the penalty functions that we defined in Table 1. The third and last objective function is thus $g_3(x) \equiv \sum_{\omega_i \in \Omega} \text{penalty}(x, \omega_i)$.

Accordingly, the constrained adversarial attack objective translates into MoEvA2 into a three-objective function to minimize with three validity conditions, that is:

- minimise $g_1(x) \equiv h(x)$ subject to $g_1(x) < t$
- minimise $g_2(x) \equiv L_p(x - x^0)$ subject to $g_2(x) \leq \epsilon$
- minimise $g_3(x) \equiv \sum_{\omega_i \in \Omega} \text{penalty}(x, \omega_i)$ subject to $g_3(x) = 0$

and this three-objective function also forms the fitness function that MoEvA2 uses to assess candidate solutions.

### 4.2 Genetic Algorithm

We instantiate MoEvA2 as a multi-objective genetic algorithm, namely based on R-NSGA-III [Vesikari et al., 2018]. We describe below how we specify the different components of this algorithm. It is noteworthy, however, that our general approach is not bound to R-NSGA-III. In particular, the three-objective function described above can give rise to other search-based approaches for constrained adversarial attacks.
**Population initialization.** The algorithm first initializes a population $P$ of $L$ solutions. Here, an individual represents a particular example that MoEvA2 has produced through successive alterations of a given original example $x$. We specify that the initial population comprises $L$ copies of $x$. The reason we do so is that we noticed, through preliminary experiments, that this initialization was more effective than using random examples. This is because the original input inherently satisfies the constraints, which makes it easier to alter it into adversarial inputs that satisfy the constraints as well.

**Population evolution.** MoEvA2 generates new individuals with two-point binary crossover. MoEvA2 selects the parents with a binary tournament over the Pareto dominance, and mutates the mutable features of the children using polynomial mutation. MoEvA2 uses non-dominance sorting based on our three objective functions to determine which individuals it keeps for the next generation. We provide the details of the algorithms in Appendix B.

## 5 Experimental Settings

### 5.1 Datasets and Constraints

Because images are devoid of constraints and fall outside the scope of our framework, we evaluate C-PGD and MoEvA2 on four datasets coming from inherently constrained domains. These datasets bear different sizes, features, and types (and number) of constraints. We evaluate both neural networks (NN) and random forest (RF) classifiers. More details about datasets and models in Appendix C.

**LCLD.** It is inspired by the Lending Club Loan Data [Kaggle, 2019]. Therein, examples are credit requests that can be accepted or rejected. We trained a neural network and a random forest that both reach an AUROC score of 0.72. We have identified constraints that include 94 boundary conditions, 19 immutable features, and 10 feature relationship constraints (3 linear, 7 non-linear). For example, the installment ($I$), the loan amount ($L$), the interest rate ($R$) and the term ($T$) are linked by the relation $I = L \times R \times (1 + R)^T / ((1 + R)^T - 1)$.

**CTU-13.** It is a feature-engineered version of CTU-13, proposed by [Chernikova and Oprea, 2019]. It includes a mix of legit and botnet traffic flows from the CTU campus. We trained a neural network and a random forest to classify legit and botnet traffic, which both achieve an AUROC of 0.99. We identified 324 immutable features and 360 feature relationship constraints (326 linear, 34 non-linear). For example, the maximum packet size for TCP ports should be 1500 bytes.

**Malware.** It comprises features extracted from a collection of benign and malware PE files [Aghakhani et al., 2020]. We trained a random forest with an AUROC of 0.99. We identified 88 immutable features and 7 feature relationship constraints (4 linear, 3 non-linear). For example, the sum of binary features set to 1 that describe API imports should be less than the value of the feature api_nb, which represents the total number of imports on the PE file.

**URL.** It comes from [Hannousse and Yahiouche, 2021] and contains a set of legitimate or phishing URLs. The random forest we use has an AUROC of 0.97. We have identified 14 relation constraints between the URL features, including 7 linear constraints (e.g. hostname length is at most equal to URL length) and 7 are if-then-else constraints.

### 5.2 Experimental Protocol and Parameters

In all datasets, a typical attack would be interested in fooling the model to classify a malicious class (rejected credit, botnet, malware, and phishing URL) into a target class (accepted, legit, benign, and legit URL). By convention, we denote by 1 the malicious class and by 0 the target class.

We evaluate the success rate of the attacks on the trained models using, as original examples, a subset of the test data from class 1. In LCLD we take 4000 randomly selected examples from the candidates, to limit computation cost while maintaining confidence in the generalization of the results. For CTU-13, Malware, and URL, we use respectively all 389, 1308, and 1129 test examples that are classified in class 1.

Since all datasets comprise binary, integer, and continuous features, we use the $L_2$ distance to measure the perturbation between the original examples and the generated examples.

We detail and justify in Appendices C and D the attack parameters including perturbation threshold $\epsilon$, the number of generations and population size for the genetic algorithm attack, and the number of iterations for the gradient attack.

## 6 Experimental Results

### 6.1 Attack Success Rate

Table 2 shows the success rate of PGD, PGD + SAT, C-PGD, and MoEvA2 on the two neural networks that we have trained on LCLD and CTU-13; and the success rate of the Papernot attack and MoEvA2 on the random forests that we have trained on each dataset. More precisely, we use the extension
of the original Papernot attack [Papernot et al., 2016] that [Ghamizi et al., 2020] proposed to make this attack applicable to random forests.

PGD and PGD + SAT fail to generate any constrained adversarial examples. The problem of PGD is that it fails to satisfy the domain constraints. While the use of a SAT solver fixes this issue, the resulting examples are classified correctly. C-PGD can create LCLD examples that satisfy the constraints and examples that the model misclassifies, yielding an actual success rate of 9.85%. On the CTU-13 dataset, however, the attack fails to generate any constrained adversarial examples. The reason is that CTU-13 comprises 360 constraints, which translates into as many new terms in the function of which C-PGD backpropagates through. As each function contributes with a diverse, non-co-linear, or even opposed gradients, this problem of PGD backpropagates through. As each function contributes with a diverse, non-co-linear, or even opposed gradients, this ultimately hinders the attack. Similar phenomena have been observed in multi-label [Song et al., 2018] and multi-task models [Ghamizi et al., 2021]. By contrast, MoEvA2, which enables a global exploration of the search space, is successful for 97.48% and 100% of the original examples, respectively.

MoEvA2 also manages to create feasible adversarial examples on the random forest models, with a success rate ranging from 5.41% to 41.51%. This indicates that our attack remains effective on such ensemble models, including with other datasets. Like PGD, the Papernot attack – unaware of constraints – cannot produce a single feasible example on LCLD, CTU-13, and Malware, whereas it has a low success rate (8.50%) on URL compared to MoEvA2 (31.89%).

**Conclusion**: While adversarial attacks unaware of domain constraints fail, incorporating constraint knowledge as an attack objective enables the successful generation of constrained adversarial examples.

### 6.2 Adversarial Retraining

We, next, evaluate if adversarial retraining is an effective means of reducing the effectiveness of constrained attacks.

We start from our models trained on the original training set. We generate constrained adversarial examples (using either C-PGD or MoEvA2) from original training examples that each model correctly classifies in class 1. To enable a fair comparison of both methods, for the LCLD (NN), we use only the original examples for which C-PGD and MoEvA2 could both generate a successful adversarial example. In all other cases, we do not apply this restriction, since MoEvA2 is the only technique that is both applicable and successful. While MoEvA2 returns a set of constrained examples, we only select the individual that maximizes the confidence of the model in its (wrong) classification, similarly to C-PGD that maximizes the model loss.

Regarding the perturbation budget, we follow established standards [Carlini et al., 2019] and provide the attack with an \( \epsilon \) budget 4 times larger than the defense.

Table 3 (middle rows) shows the results for the neural networks, and Table 4 (second row) for the random forests. Overall, we observe that adversarial retraining remains an effective defense against constrained attacks. For instance, on LCLD (NN) adversarial retraining using MoEvA2 drops the success rate of C-PGD from 9.85% to 2.70%, and its own success rate from 97.48% to 85.20%. The fact that MoEvA2 still works suggests, however, that the large search space that this search algorithm explores preserves its effectiveness. By contrast, on CTU-13 (NN), we observe that the success rate of MoEvA2 drops from 100% to 0.8% after adversarial retraining with the same attack.

**Conclusion**: Adversarial retraining remains an effective defense against constrained adversarial attacks.

### 6.3 Defending With Engineered Constraints

We hypothesize that an alternative way to improve robustness against constrained attacks is to augment \( \Omega \) with a set of engineered constraints – in particular, non-convex constraints.

To verify this, we propose a systematic method to add engineered constraints, and we evaluate the effectiveness of this novel defense mechanism.

To define new constraints, we first augment the original data with new features engineered from existing features. Let \( \hat{f} \) denote the mean value of some feature of interest \( f \) over the training set. Given a pair of feature \( (f_1, f_2) \), we engineer a binary feature \( f_e \) as

\[
f_e(x) \equiv (\hat{f}_1 \leq x_1) \oplus (\hat{f}_2 \leq x_2)
\]

where \( x_i \) is the value of \( f_i \) in \( x \) and \( \oplus \) denotes the exclusive or (XOR) binary operator. The comparison of the value of the input \( x \) for a particular features \( f_i \) with the mean \( \hat{f}_i \) allows us to handle non-binary features while maintaining a uniform distribution of value across the training dataset. We use the XOR operator to generate the new feature as this operator is...
not differentiable. We then, introduce a new constraint that the value of \( f_e \) should remain equal to its original definition. That is, we add the constraint

\[
\omega_e \equiv (f_e(x) = (\hat{f}_1 \leq x_1) \oplus (\hat{f}_2 \leq x_2))
\]

The original examples respects these new constraints by construction. In other words, for an adversarial attack to be successful, the attack should modify \( f_e \) if the modifications it applied to \( \hat{f}_1 \) and \( \hat{f}_2 \) would imply a change in the value of \( f_e \).

To avoid combinatorial explosion, we add constraints only on pairs of the most mutable features. We measure importance with the approximation of Shapley value [Shrikumar et al., 2017], an established explainability method. In the end, we consider a number \( M \) of pairs such that \( M = \arg \max_x \binom{\tilde{x}}{2} < \frac{N}{4} \) where \( N \) is the total number of features.

As a preliminary sanity check, we verified that constraint augmentation does not penalize clean performance and confirmed that the augmented models keep similar performance.

To evaluate this constraint augmentation defense, we use the same protocol as before, except that the models are trained on the augmented set of features. That is, we assume that the attacker has knowledge of the added features and constraints.

We show the results in Table 3 and 4 (third row). Constrained augmentation nullifies the low success rate of C-PGD on LCLD – the gradient-based attack becomes unable to satisfy the constraints. Our defense also decreases significantly the success rate of MoEvA2 in all cases except the CTU-13 random forest. For instance, it drops from 97.48% to 80.43% for LCLD NN, and from 100% to 0% on CTU-13 NN.

We assess the effect of constraint augmentation and adversarial retraining more finely and show, in Figure 1, the success rate of MoEvA2 on the LCLD neural network over the number of generations. Compared to the undefended model, both constrained augmentation and adversarial retraining (using MoEvA2) lower the asymptotic success rate. Moreover, the growth of success rate is much steeper for the undefended model. For instance, MoEvA2 needs ten times more generations to reach the same success rate of 84% against the defended models than against the undefended model (100 generations versus 12 generations). Adversarial retraining using C-PGD is less effective: while it reduces the success rate in the earlier generations, its benefits diminishes as the MoEvA2 attack runs for more generations.

**Conclusion**: Constraint augmentation is an effective alternative defense against constrained adversarial attacks. The benefits of both defense mechanisms against MoEvA2 are achieved as soon as the earliest generations and persist throughout the attack process.

### 6.4 Combining Defenses

We investigate whether the combination of constraint augmentation with adversarial retraining yields better results. A positive answer would indicate that constraint augmentation and adversarial retraining have complementary benefits.

We add to the models the same engineered constraints as we did previously. We also perform adversarial retraining on the augmented models, using all adversarial examples that MoEvA managed to generate on the training set. Then, we attack the defended models using MoEvA applied to the test set. For a fair comparison with adversarial retraining, we also apply this defense without constraint augmentation, using the same number of examples. We do not experiment with C-PGD, which was already ineffective when only one defense was used. Neither do we consider the datasets for which one defense was enough to fully protect the model.

Tables 3 and 4 (last rows) present the results. On LCLD (NN), the combined defenses drops the attack success rate from 97.48% (on a defenseless model) to 77.23%, which better than adversarial retraining (82.00%) and constraint augmentation (80.43%) applied separately. On the RFs, the combination either offers additional reductions in attack success rate compared to the best individual defense (LCLD and URL) or has negligible effects (CTU-13 and Malware).

**Conclusion**: Constraint augmentation and adversarial retraining are two effective defense strategies that have complementary effects. Compared to their separate application, the combination can decrease the attack success rate by up to 5%.

### 7 Conclusion

We proposed the first generic framework for adversarial attacks under domain-specific constraints. We instantiated our framework with two methods: one gradient-based method that extends PGD with multi-loss gradient descent, and one that relies on multi-objective search. We evaluated our methods on four datasets and two types of models. We demonstrated their unique capability to generate constrained adversarial examples. In addition to adversarial retraining, we proposed and investigated a novel defense strategy that introduces engineered non-convex constraints. This strategy is as effective as adversarial retraining. We hope that our approach, algorithms, and datasets will be the starting point of further endeavor towards studying feasible adversarial examples in real-world domains that are inherently constrained.
Acknowledgments

Salijona Dyrmishi is supported by the Luxembourg National Research Funds (FNR) AFR Grant 14585105.

References

[Aghakhani et al., 2020] Hojjat Aghakhani, Fabio Gritti, Francesco Mecca, Martina Lindorfer, Stefano Orlonti, 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.

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

[Carlini et al., 2019] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin. On evaluating adversarial robustness. ArXiv, abs/1902.06705, 2019.

[Chernikova and Oprea, 2019] Alesia Chernikova and Alina Oprea. Fence: Feasible evasion attacks on neural networks in constrained environments. arXiv preprint arXiv:1909.10480, 2019.

[Dalvi et al., 2004] Nilesh Dalvi, Pedro Domingos, Sumit Sahngai, and Deepak Verma. Adversarial classification. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 99–108, 2004.

[Du et al., 2020] Tianyu Du, Shouling Ji, Jinfeng Li, Qinchen Gu, Ting Wang, and Raheem Beyah. Sirenaattack: Generating adversarial audio for end-to-end acoustic systems. In Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, pages 357–369, 2020.

[Erdenem et al., 2021] Ecenaz Erdemir, Jeffrey Bickford, Luca Melis, and Sergul Aydore. Adversarial robustness with non-uniform perturbations. arXiv preprint arXiv:2102.12002, 2021.

[Ghamizi et al., 2020] Salah Ghamizi, Maxime Cordy, Martin Gubri, Mike Papadakis, Andrey Boyostov, Yves Le Traon, and Anne Goujon. Search-based adversarial testing and improvement of constrained credit scoring systems. In Proc. of ESEC/FSE ’20, pages 1089–1100, 2020.

[Ghamizi et al., 2021] Salah Ghamizi, Maxime Cordy, Mike Papadakis, and Yves Le Traon. Adversarial robustness in multi-task learning: Promises and illusions. arXiv preprint arXiv:2110.15053, 2021.

[Gurobi Optimization, LLC, 2022] Gurobi Optimization, LLC. Gurobi Optimization Reference Manual, 2022. https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/9.5/reftut.pdf.

[Hannousse and Yahioiche, 2021] Abdelhakim Hannousse and Salima Yahioiche. Towards benchmark datasets for machine learning based website phishing detection: An experimental study. Engineering Applications of Artificial Intelligence, 104:104347, 2021.

[Kaggle, 2019] Kaggle. All Lending Club loan data, 2019.

[Kurakin et al., 2016] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236, 2016.

[Li et al., 2020] 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.

[Mode and Hoque, 2020] Gautam Raj Mode and Khaza Anuarul Hoque. Crafting adversarial examples for deep learning based prognostics (extended version). arXiv preprint arXiv:2009.10149, 2020.

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

[Pierazzi et al., 2020] 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.

[Sadegh and Larsson, 2019] Meysam Sadeghi and Erik G Larsson. Physical adversarial attacks against end-to-end autoencoder communication systems. IEEE Communications Letters, 23:847–850, 2019.

[Sheatsley et al., 2020] Ryan Sheatsley, Nicolas Papernot, Michael Weissman, Gunjan Verma, and Patrick McDaniel. Adversarial examples in constrained domains. arXiv preprint arXiv:2011.01183, 2020.

[Shrikumar et al., 2017] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In International Conference on Machine Learning, PMLR, 2017.

[Song et al., 2018] Qingquan Song, Haifeng Jin, Xiao Huang, and Xia Hu. Multi-label adversarial perturbations. In 2018 IEEE International Conference on Data Mining (ICDM), pages 1242–1247. IEEE, 2018.

[Szegedy et al., 2013] 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.

[Tian et al., 2020] Yunzhe Tian, Yingdi Wang, Endong Tong, Wenhua 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.

[Vesikar et al., 2018] Yash Vesikar, Kalyanmoy Deb, and Julian Blank. Reference point based nsga-iii for preferred solutions. In 2018 IEEE symposium series on computational intelligence (SSCI), pages 1587–1594. IEEE, 2018.

[Yefet et al., 2020] Noam Yefet, Uri Alon, and Eran Yahav. Adversarial examples for models of code. Proceedings of the ACM on Programming Languages, 4:1–30, 2020.