TOWARDS ROBUST ADVERSARIAL EXAMPLES FOR DEEP NEURAL NETWORKS

JÖRG RAMBAU[1], RONAN RICHTER[1,*]

[1]*Chair for Economathematics, Faculty of Mathematics, Physics and Computer Sciences, University of Bayreuth, Germany*

Abstract. In this paper, we show two methods to compute sampling-robust adversarial examples (AEs) for deep neural networks with rectilinear units (DNNs). Both methods use an adjustable robust counterpart of a MILP model by Fischetti an Jo. They rely on new uncertainty sets in (pseudo-)metric spaces of DNNs with identical structure and compact inputs. One method (the inner method) needs full information on weights and biases of a nominal DNN after training. The other one (the outer method) only needs full information on the training data and the training method used. We compare the two methods in experiments on DNNs classifying small fashion images according to the type of apparel shown. While the inner method generates AEs that are only robust w.r.t. very mild retraining of a DNN, the outer method leads to AEs that are robust w.r.t. retraining from scratch on the same training data. The outer approach can therefore in principle be used for grey-box attacks of DNNs with no knowledge on internal parameters after training.

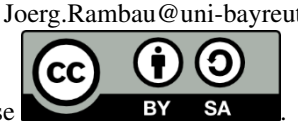
Keywords. Adversarial Examples; Deep Neural Networks; Robust Optimization; Mixed-Integer Optimization.

1. INTRODUCTION

In this paper, exact optimization methods to generate adversarial examples (AE) for a given ReLU Deep Neural Network (DNN) are extended using methods of *robust* optimization in order to obtain AEs that are robust w.r.t. "small" changes in the DNN (definitions and details below). Previous exact optimization methods need full information on the given DNN (all weights and biases) and, therefore, enable only *white-box attacks*. Our proposed method only needs information on the training data and the possible training methods that were used to train the DNN to be attacked (no weights and biases), which can be seen as a *grey-box attack*. Our new idea is the following two-step approach: first, formulate an adjustable robust counterpart w.r.t. to an uncertainty set in some space of DNNs; second, solve this (intractable) counterpart approximatively by sampling in the uncertainty set.

While there are numerous approaches for leveraging machine learning techniques for solving optimization problems [8], classical mathematical optimization methods have in turn been used to improve or analyze machine learning structures [14]. One direction of the second class of research is embedding DNNs in mixed-integer linear programs [3, 4]. More specifically, [13]

*Corresponding author.
E-mail addresses: Ronan.Richter@uni-bayreuth.de, Joerg.Rambau@uni-bayreuth.de.

This is a preprint under the creative commons license  .
The final publication is available at Journal of Applied and Numerical Optimization, via https://doi.org/10.23952/jano.7.2025.3.02.

and [21] have independently used methods from mixed-integer linear optimization (MILP) to evaluate the vulnerability of DNNs to *adversarial examples*.

The notion of an adversarial example (AE) was coined in [20] as follows. For a nominal input with known true classification value, an AE is an input with "small" distance to the nominal input and different classification value. Sometimes, a minimal misclassification probability (output activation) is required for an AE. With MILP-model in [13], e.g., one can exactly minimize the distance of an AE from the nominal input for DNNs bases on rectilinear units (ReLU). This minimal distance can be interpreted as the robustness *of the DNN against perturbations of a nominal input*. For an overview other methods for generating adversarial examples see [2]. Note that only exact optimization methods (i.e., methods generating reliable lower and upper bounds) can, e.g., deal with the *non-existence* of adversarial examples at a given distance, which would prove that a DNN is robust in this sense. The draw-back of these known exact MILP-approaches is that full information on the trained parameters (withs and biases) of a DNN is needed and that the computational effort is quite large even for small DNN-structures.

For an AE it is not guaranteed that it stays an AE if the DNN changes a little. For example, the information about weights and biases might be uncertain. For the transferability of AEs to certain other DNNs see [16]. There, robust optimization appears only in the context of robustification of a DNN w.r.t. to perturbed AEs or robustification of an AE w.r.t. to perturbations of itself (see also [5]). However, our goal is to determine adversarial examples that are robust to changes of the DNN. Such examples could give insights to more general limits of DNNs or a certain class of DNNs, i.e. DNNs with a given structure. Furthermore, taking more than one DNN into account can increase the transferability of adversarial examples (see, e.g., [17]). However, as far as we know, the presented methods for generating adversarial examples for multiple DNNs so far are not exact.

In this manuscript, we combine the proposed mixed-integer formulation of [13] with robust optimization techniques [6] in order to generate robustness *of an AE against perturbations of the DNN*. In particular, using *adjustable robustness* [7], we formally design a model for determining adversarial examples that lead to misclassifications for an uncertainty set of DNNs. Since this model contains binary variables in the second stage, the standard methods for exactly solving adjustable robust optimization problems cannot be applied. For an overview of solution methods for adjustable robust programs with integer second stage variables, see [24, Sec. 6]. The exact methods for such programs [10, 19] based on splitting the uncertainty set, require the uncertain coefficient matrices to be linear in the uncertain parameter, which is not the case for our program where the uncertain parameter is a neural network. Therefore, we suggest to use extensive formulations based on a finite number of samples from the uncertainty set (see [9]). AEs generated this way will be called *sample-robust*.

We investigate two possible types of uncertainty sets that are both ε-neighborhoods in some metric space of DNNs: one based on *perturbations of the weights and biases* of a nominal DNN (*inner neighborhood*), and one based on *perturbations of the resulting classification function* in the space of all such functions generated by DNNs of identical inner structure (layers and neurons) that have been trained on a given set of training data by a given set of training methods (*outer neighborhood*). While the computation of an AE that is sample-robust w.r.t. an inner neighborhood set (*inner-sample robust AE*) still needs the weights and biases of a nominal

DNN, the computation of an AE that is sample-robust w.r.t. an outer neighborhood (*outer-sample robust AE*) is independent from any weights and biases. Sampling in the latter case consists of *simulation*, i.e., of generating DNNs from scratch using the training data and the training methods used for the DNN to be attacked and dismissing DNNs that are not in the outer neighborhood.

Table 2 of our experimental results shows that DNNs representing similar classification functions do not necessarily have similar internal trained parameters. Thus, the inner neighborhood may miss many DNNs with similar classification properties. Therefore, it can be expected that inner-sample robustness need not be similar to outer-sample robustness. Indeed, our computational results show on examples from the MNIST-fashion dataset that the two approaches differ: while the inner-sample robustness can be achieved easily even without or any kind of robust optimization, the outer-sample robustness can only be achieved by outer-sample robust AEs. In all cases, surprisingly few samples are sufficient to obtain a reasonable level of robustness.

The contributions of this paper are the following:

- We define two types of robust adversarial examples by defining two (pseudo-)metric spaces on the set of all deep neural networks.
- We compare the two types of robust adversarial examples on a standard data set of small fashion images from ten classes of apparel.
- We show that even a small number of samples from a suitable uncertainty set can substantially increase the robustness of the generated adversarial example.
- We show how the so-called outer-sampling robustness can be utilized for a grey-box attack of a DNN with unknown trained parameters but with known training data and training method.

The rest of the paper is organized as follows. In Section 2 we recapitulate the notions for DNNs and Robust Optimization that are important for this paper. In Section 3 we introduce the two types of metric spaces on the set of all DNNs that lead to suitable uncertainty sets for the robust-optimization approach. Section 4 presents our adjustable robust optimization model and some details on the sampling method used to solve it. The experimental results are provided in Section 5, before our conclusions are drawn in Section 6.

2. PRELIMINARIES

In this paper, DNNs are formally considered as parametrized functions (for more explanations, see, e.g., [13]):

Definition 2.1 (DNN)**.** A *ReLU Deep Neural Network Structure (DNN-structure) with* $K+1$ *layers of sizes* $n_0, n_1, \ldots, n_K$ is a function

$$f\colon \left\{ \begin{array}{rcl} \mathbb{R}^{n_0} \times \mathbb{R}^{n_0 \times n_1 + \cdots + n_{K-1} \times n_K} \times \mathbb{R}^{n_1 + \cdots + n_K} & \to & \mathbb{R}^{n_K} \\ (x, w, b) & \mapsto & y, \end{array} \right.$$

where

$$\begin{aligned} x_j^0 &= x_j, \\ y_j &= x_j^K, \\ x_j^k &= \max\left\{\sum_{i=1}^{n_{k-1}} w_{ij}^{k-1} x_i^{k-1} + b_j^k, 0\right\} \text{ for } k = 1,2,\ldots,K. \end{aligned}$$

The vector x is the *input data*, the vector y is the *output data*, the w are the *weights*, and the b are the *biases* of the DNN-structure.

A *ReLU Deep Neural Network (DNN)* $N = (f,w,b)$ is a DNN-structure f together with given weights w and biases b, resulting in a function of the following form:

$$f_{w,b}\colon \left\{ \begin{array}{rcl} \mathbb{R}^{n_0} & \to & \mathbb{R}^{n_K} \\ x & \mapsto & y := f(x,w,b). \end{array} \right.$$

The process of generating weights and biases for a DNN-structure to obtain a DNN is called *training*.

The *standard softmax function* is the following function:

$$\text{softmax}\colon \left\{ \begin{array}{rcl} \mathbb{R}^{n_K}_{\geq 0} & \to & [0,1]^{n_K} \\ y & \mapsto & \left(\frac{\exp\left(y_1\right)}{\sum_{i=1}^{n_K} \exp\left(y_{n_i}\right)}, \ldots, \frac{\exp\left(y_{n_K}\right)}{\sum_{i=1}^{n_K} \exp\left(y_{n_i}\right)} \right). \end{array} \right.$$

The *classification probability function* of a DNN $N = (f,w,b)$ is the composite function $p_{w,b} := \text{softmax} \circ f_{w,b}$, and the *classification function* of a DNN $N = (f,w,b)$ is the function

$$c_N\colon \left\{ \begin{array}{rcl} \mathbb{R}^{n_0} & \to & \{1,2,\ldots,n_K\} =: \mathscr{C} \\ x & \mapsto & \text{argmax}\big(p_{w,b}(x)\big). \end{array} \right.$$

For the training of a DNN-structure weights and biases are adapted to known pairs (x,y) of input and output data (the *training data*) using some *training method* that aims to find weights w and biases b that minimize some norm of $f_{w,b}(x) - y$. The classification functions of DNNs are often used for classification problems. The *accuracy* of (the classification function of) a DNN of a classification problem on known pairs (x,y) of input and output data not in the training data (the *test data*) is the percentage of pairs (x,i) in the test data with $c_N(x) = i$.

For the purpose of judging whether a possible input that is neither contained in the training data nor in the test data, can be considered as incorrectly classified by a DNN, we need the "true" classification of this input. We, therefore, define the true classifier of a classification problem that allows us to define rigorously what an adversarial example is supposed to be.

Definition 2.2. Let $\mathscr{X}$ be the input domain of a classification problem with classes $\mathscr{C}$. The *true classifier* is a function $T^\star\colon \mathscr{X} \to \mathscr{C}$ that maps each possible input $x \in \mathscr{X}$ to a class $T^\star(x) \in \mathscr{C}$. The output of $T^\star(x)$ is called the *true class of* x. Given a DNN N, an input $\bar{x}$ with $c_N(\bar{x}) = T^\star(\bar{x})$, and a distance $\delta > 0$, an *adversarial example* for N w.r.t. $\bar{x}$ and δ is an input x with $\|\bar{x} - x\| < \delta$ so that $c_N(x) \neq T^\star(x) = T^\star(\bar{x})$.

The following MILP model for generating adversarial examples was presented in [13]:

$$
\begin{array}{rll}
\min\limits_{x,s,z,d} \sum\limits_{j=0}^{n_0} d_j & & \\
\text{s.t. } -d_j \leq x_j^0 - \bar{x}_j^0 \leq d_j & & \forall j = 1,\ldots,n_0 \\
d_j \geq 0 & & \forall j = 1,\ldots,n_0 \\
\bar{l}_j^0 \leq x_j^0 \leq \bar{u}_j^0 & & \forall j = 1,\ldots,n_0 \\
\sum\limits_{i=1}^{n_{k-1}} w_{i,j}^{k-1} x_i^{k-1} + b_j^k = x_j^k - s_j^k & & \forall k = 1,\ldots,K \, \forall j = 1,\ldots,n_k \\
x_j^k, s_j^k \geq 0 & & \forall k = 1,\ldots,K \, \forall j = 1,\ldots,n_k \\
z_j^k \in \mathbb{B} & & \forall k = 1,\ldots,K \, \forall j = 1,\ldots,n_k \\
z_j^k = 1 \rightarrow x_j^k \leq 0 & & \forall k = 1,\ldots,K \, \forall j = 1,\ldots,n_k \\
z_j^k = 0 \rightarrow s_j^k \leq 0 & & \forall k = 1,\ldots,K \, \forall j = 1,\ldots,n_k \\
x_{\text{adv}}^K \geq (1+\gamma) x_j^K & & \forall j \in \{1,\ldots,n_K\} \setminus \{\text{adv}\}
\end{array}
$$

In this model (we call it the *FJ-model*), $\bar{l}$ and $\bar{u}$ are lower and upper bounds on the components of the input (application dependent). The data $\bar{x}$ denotes the correctly classified input (*true class*) and the variable x_{adv}^K is the output activation for a fixed incorrect class (*target class*). Moreover, the parameter γ measures by how much the activation for the target class must be larger than the activation for the true class on input x. This x is the variable for the adversarial example input. Furthermore, d is an auxiliary variable used to measure the L_1-distance of x and $\bar{x}$. The objective function models that we seek for an input x with minimal L_1-distance to a correctly classified input $\bar{x}$. The constraints guarantee that the wrong classification has at least a given "confidence". In order to model the ReLU-propagation of activation values in the inner layers, the binary auxiliary variables z and the slack variables s. The variable-dependent indicator constraints (with a "$\rightarrow$") denote a case distinction between different constraints that should hold when z_j^k is 0 or 1, resp. It is a standard-technique of MILP to translate these into feasible MILP-constraints by the so-called "Big-M-method".

In this model it is implicitly assumed that a small distance of inputs implies that the true classification does not change. This is a matter of human interpretation, though. Note that all weights and biases of the DNN are needed as input parameters of the FJ-model.

The main reason for the hope that there must be adversarial examples that are robust against small changes in the weights and biases of the given DNN is the following result.

Remark 2.1 ([22])**.** Any DNN-structure f is Lipschitz continuous.

For finding robust adversarial examples we seek to formulate and solve a suitable robust counterpart of the FJ-model. Since the inner activation values are completely arbitrary (as long as they exist), the concept of *adjustable robust counterparts* is appropriate.

The concept of adjustable robust optimization has been introduced in [7]. We will briefly describe the general idea following the notation of [15]. As usual in robust optimization, we are given an uncertainty set $\mathscr{U}$, consisting of scenarios $\xi \in \mathscr{U}$, representing different realizations

of uncertain data. The goal is to solve the family of problems $(P(\xi), \xi \in \mathscr{U})$ with

$$\left|\begin{array}{l} \min f(x,y,\xi) \\ \text{s.t. } F(x,y,\xi) \leq 0 \\ \qquad\qquad x \in X \\ \qquad\qquad y \in Y \end{array}\right| \qquad (P(\xi))$$

with functions $F(\cdot,\cdot,\xi)\colon \mathbb{R}^{d_1+d_2} \to \mathbb{R}^m$ and $f(\cdot,\cdot,\xi)\colon \mathbb{R}^{d_1+d_2} \to \mathbb{R}$ for all $\xi \in \mathscr{U}$. The information structure is so, that the variable x must be set independently of the scenario $\xi \in \mathscr{U}$ (the *non-anticipative variables* or *first-stage variables* or *here-and-now variables*), whereas y can be chosen individually dependent on ξ (the *recourse variables* or *second-stage variables*, or *wait-and-see variables*, or *adjustable variables*). The set of feasible solutions in scenario ξ is

$$\mathscr{F}(\xi) := \left\{(x,y) \in \mathbb{R}^{d_1+d_2} \middle| F(x,y,\xi) \leq 0\right\}$$

while the set of feasible first-stage adjustable robust solutions is denoted as

$$\mathscr{F} := \{x \in X \mid \forall \xi \in \mathscr{U}\ \exists y \in Y : F(x,y,\xi) \leq 0\}.$$

The goal of adjustable robust optimization is to find the feasible first-stage solution with the minimal objective value in the worst case, i.e., solving the problem

$$\min_{x \in \mathscr{F}} \sup_{\xi \in \mathscr{U}} \inf_{(x,y) \in \mathscr{F}(\xi)} f(x,y,\xi).$$

This optimization problem is called the *adjustable robust counterpart of the uncertain optimization problem* $(P(\xi), \xi \in \mathscr{U})$. For a given *nominal parameter* $\bar{\xi} \in \mathscr{U}$ the optimization problem $P(\bar{\xi})$ is called the *nominal problem of* $(P(\xi), \xi \in \mathscr{U})$.

For a finite uncertainty set $\mathscr{U}$, the adjustable robust counterpart can be written as follows:

$$\left|\begin{array}{ll} \min \sup_{\xi \in \mathscr{U}} f(x,y(\xi),\xi) & \\ \quad \text{s.t. } F(x,y(\xi),\xi) \leq 0 & \forall \xi \in \mathscr{U} \\ \qquad\qquad x \in X & \\ \qquad\qquad y(\xi) \in Y & \forall \xi \in \mathscr{U} \end{array}\right|$$

This is called the *extensive form* of the adjustable robust counterpart. It usually is an optimization problem of the same type as the nominal problem but larger. Hence, it can be solved by the same techniques as the nominal problem. There is no general exact solution method known for the adjustable robust counterpart for infinite uncertainty sets. One class of problems for which no exact solution algorithm is known is an adjustable robust counterpart with binary recurse variables that have coefficients that are non-linear in ξ. In such cases, an approximate method to solve a robust adjustable counterpart is to replace the uncertainty set by a finite number of samples ([9]) and solve the corresponding extensive form of the adjustable robust counterpart exactly.

3. Metric Spaces of DNNs

In this section, we define two metric spaces on the set of all DNNs with identical DNN-structure. The plan is to generate uncertainty sets for an adjustable robust counterpart based on

- a nominal DNN $\bar{N} := (f, \bar{w}, \bar{b})$, representing some known DNN correctly classifying input $\bar{x}$,
- an ε-neighborhood around $\bar{N}$ in some metric space of DNNs, representing all DNNs for which an adversarial example x close to $\bar{x}$ for $\bar{N}$ shall stay an adversarial example.

Definition 3.1. For a DNN-structure f, let $\mathscr{N} := \{N | N = (f, w, b)\}$ be the set of all DNNs with DNN-structure f.

From now on we assume that the set $\mathscr{X}$ of possible inputs is compact. This is, e.g., the case for all pixel images with a given resolution and bit-depth. Our benchmark example will be 8-bit gray-scale images with 28×28 pixels.

The first metric space on $\mathscr{N}$ that comes to mind is the space of all weights and biases with some vector norm. We make the following choice. The goal we keep in mind is to obtain an adjustable robust counterpart of the FJ-model that is "as tractable as possible".

Definition 3.2. The *inner distance* of two DNNs $N = (f, w, b)$ and $N' = (f, w', b')$ in $\mathscr{N}$ is

$$d^{\text{in}}(N, N') := \left\| \begin{pmatrix} w \\ b \end{pmatrix} - \begin{pmatrix} w' \\ b' \end{pmatrix} \right\|.$$

The *inner ε-neighborhood* of a DNN $\bar{N}$ is a set of the form $U_\varepsilon^{\text{in}}(\bar{N}) := \{N | d^{\text{in}}(N, \bar{N}) < \varepsilon\}$. The *inner-DNN-space* is the metric space $\mathscr{N}^{\text{in}} := (\mathscr{N}, d^{\text{in}})$.

Robustness of an AE w.r.t. to the inner neighborhood means that the AE stays an AE even when in $\bar{N}$ small modifications of the weights and biases are initiated. Such a small perturbation of weights and biases may happen when an otherwise known DNN is retrained on very few additional training data.

However, if a DNN is trained from scratch with the same training data and training method as $\bar{N}$ the usual stochastic nature of the training methods can lead to completely different weights and biases although the resulting classification function maybe very similar. Table 2 shows this for our benchmark example. Therefore, in the more sophisticated approach below we focus on the DNN representing classification probabilities.

Definition 3.3. The *outer distance* of two DNNs $N = (f, w, b)$ and $N' = (f, w', b')$ representing classification probabilities $p_N := p_{w,b}$ and $p_{N'} := p_{w',b'}$ respectively, is defined as

$$d^{\text{out}}(N, N') := \int_{\mathscr{X}} \left| p_N(x) - p_{N'}(x) \right| \mathrm{d}x.$$

The *outer ε-neighborhood* of a DNN $\bar{N}$ is $U_\varepsilon^{\text{out}}(\bar{N}) := \{N | d^{\text{out}}(N, \bar{N}) < \varepsilon\}$. The *outer-DNN-space* is the metric space $\mathscr{N}^{\text{out}} := (\mathscr{N}, d^{\text{out}})$.

Since DNNs are Lipschitz continuous (see Remark 2.1), $d^{\text{out}}(N, N')$ defines a metric on the space of DNNs with input domain $\mathscr{X}$.

Unfortunately, the outer distance cannot be evaluated exactly since we cannot check all inputs to evaluate the integral. With this intractability problem we again deal with sampling in the following way.

Definition 3.4. Let $\bar{X} = \{\bar{x}^0, \ldots, \bar{x}^m\} \subset \mathscr{X}$ be a fixed set of samples of possible inputs. For a DNN $N = (f, w, b)$ representing classification probabilities $p_N = p_{w,b}$ let

$$O_N := \left(p_N(\bar{x}^0), \ldots, p_N(\bar{x}^m) \right) \in \mathbb{R}^{n_K \times m}.$$

The *outer-sampling pseudo-distance* of two DNNs N and N' is then defined as

$$d^{\mathrm{osd}}(N,N') := \frac{\|O_N - O_{N'}\|}{m}.$$

The *outer-sampling ε-neighborhood* of a DNN $\bar{N}$ is $U_\varepsilon^{\mathrm{osd}}(\bar{N}) := \{N | d^{\mathrm{osd}}(N,\bar{N}) < \varepsilon\}$. The *outer-sampling-DNN-space* is the pseudometric space $\mathscr{N}^{\mathrm{osd}} := (\mathscr{N}, d^{\mathrm{osd}})$.

For a nominal DNN $\bar{N}$ and a given uncertainty parameter ε we can now define possible uncertainty sets we want to be robust against as follows.

$$\mathscr{Z}^{\mathrm{osd}}(\bar{N}) := U_\varepsilon^{\mathrm{osd}}(\bar{N}),$$
$$\mathscr{Z}^{\mathrm{in}}(\bar{N}) := U_\varepsilon^{\mathrm{in}}(\bar{N}).$$

4. A Model for Robust Adversarial Examples

The next step of our plan is to setup an adjustable robust counterpart for generating robust adversarial examples with protection level $\gamma > 0$ against some finite uncertainty set $\mathscr{Z}$. The not-anticipative variables only consist of the input and the auxiliary variables for its distance to the nominal DNN, whereas all the other activation levels and all other auxiliary variables are recourse variables. That is, we are only interested in their existence, but do no care about their values. Note that the objective function does not contain any dependence on the uncertain parameter (i.e., the DNN). Therefore, the sup inf is just a constant value.

$$\left|\begin{aligned}
&\min_{x,s,z,d} \sum_{j=0}^{n_0} d_j && \\
&\text{s.t. } -d_j \le x_j^0 - \bar{x}_j^0 \le d_j && \forall j = 1,\dots,n_0 \\
&\qquad d_j \ge 0 && \forall j = 1,\dots,n_0 \\
&\qquad \bar{l}_j^0 \le x_j^0 \le \bar{u}_j^0 && \forall j = 1,\dots,n_0 \\
&\sum_{i=1}^{n_{k-1}} w_{i,j}^{k-1}(\xi) x_i^{k-1}(\xi) + b_j^k(\xi) = x_j^k(\xi) - s_j^k(\xi) && \forall k = 1,\dots,K \forall j = 1,\dots,n_k \forall \xi \in \mathscr{Z} \\
&\qquad x_j^k(\xi), s_j^k(\xi) \ge 0 && \forall k = 1,\dots,K \forall j = 1,\dots,n_k \forall \xi \in \mathscr{Z} \\
&\qquad z_j^k(\xi) \in \mathbb{B} && \forall k = 1,\dots,K \forall j = 1,\dots,n_k \forall \xi \in \mathscr{Z} \\
&\qquad z_j^k(\xi) = 1 \to x_j^k(\xi) \le 0 && \forall k = 1,\dots,K \forall j = 1,\dots,n_k \forall \xi \in \mathscr{Z} \\
&\qquad z_j^k(\xi) = 0 \to s_j^k(\xi) \le 0 && \forall k = 1,\dots,K \forall j = 1,\dots,n_k \forall \xi \in \mathscr{Z} \\
&\qquad x_{\mathrm{adv}}^K(\xi) \ge (1+\gamma) x_j^K(\xi) && \forall j \in \{1,\dots,n_K\} \setminus \{\mathrm{adv}\} \forall \xi \in \mathscr{Z}
\end{aligned}\right| \tag{4.1}$$

where $\mathscr{Z}$ is a finite uncertainty set. We call this model the *adjustable robust FJ-model (ARFJ-model)*.

We would like to plug-in the uncertainty sets $\mathscr{Z}^{\mathrm{in}}(\bar{N})$ and $\mathscr{Z}^{\mathrm{osd}}(\bar{N})$, resp. However, those sets are uncountably infinite. Be aware, that the z-Variables are binary recourse variables. And other recourse variables are bound by an equality constraint. Therefore, the standard methods for solving adjustable robust programs for arbitrary uncertainty sets cannot be applied [24, Sec. 6]. Thus, for approximating solutions, we restrict ourself to discrete samples of the uncertainty set. As presented in [9], discrete sampling of the uncertainty set leads to a two-stage problem in

extensive form as it is used in stochastic programming [11], that may be solved with standard MILP-methods. For further reference, we denote random samples of size α as follows:

$$\mathscr{Z}_\alpha^{\text{in}}(\bar{N}) := \left\{ \xi^1, \ldots, \xi^\alpha \middle| \xi^i \in \mathscr{Z}^{\text{in}}(\bar{N}) \ \forall i = 1, \ldots, \alpha \right\} \cup \{\bar{N}\},$$
$$\mathscr{Z}_\alpha^{\text{osd}}(\bar{N}) := \left\{ \xi^1, \ldots, \xi^\alpha \middle| \xi^i \in \mathscr{Z}^{\text{osd}}(\bar{N}) \ \forall i = 1, \ldots, \alpha \right\} \cup \{\bar{N}\}.$$

The ARFJ-model w.r.t. the *inner-sampling uncertainty set* $\mathscr{Z}_\alpha^{\text{in}}(\bar{N})$ is called the *ISFJ-model*. The ARFJ-model w.r.t. *outer-sampling uncertainty set* $\mathscr{Z}_\alpha^{\text{osd}}(\bar{N})$ is called the *OSFJ-model*. In practice, for $\mathscr{Z}_\alpha^{\text{in}}$ we uniformly draw a vector x with $\|x\| = \varepsilon$ and set $\begin{pmatrix} w \\ b \end{pmatrix} = \begin{pmatrix} \bar{w} \\ \bar{b} \end{pmatrix} + x$. For $\mathscr{Z}_\alpha^{\text{osd}}$ we include $\bar{N}$ and train networks N from scratch with the same training data by the same training method as $\bar{N}$. If $d^{\text{osd}}(N, \bar{N}) < \varepsilon$, we include N in $\mathscr{Z}_\alpha^{\text{in}}$. Otherwise we discard N. We repeat this strategy until α DNNs have been included. The weights and biases of the nominal DNN and the α sampled DNNs are then used in the OSFJ-model. Since we have full control over the sampled DNNs, this does not require any information on weights and biases of any opaque DNN except the nominal DNN $\bar{N}$. This methods seems to be specifically appealing for our purpose: if we only know the DNN-structure, the training data, and the training method but not the weights and biases of a nominal DNN, then all these sampled DNNs could have been the ones we would like to find an AE for.

Even more appealing is the fact that we could as well use the nominal DNN only as an oracle on the training data, without any knowledge of its weights and biases, only to evaluate the outer-sampling pseudodistance. Using only the α sampled DNNs in the OSFJ-model gives us the opportunity to attack an uncertainty set of DNNs without any knowledge on weights and biases that we have not generated by ourselves. This is the core of our *grey-box attack*.

In the following, an AE computed by the FJ-model is called a *nominal AE* denoted by *FJAE*, an AE computed by the ISFJ-model is called an *inner-sampling robust AE*, denoted by *IRAE*, an AE computed by the OSFJ-model is called an *outer-sampling robust AE* denoted by *ORAE*.

5. Experimental Results

In this section we experimentally compare our IRAEs and ORAEs with the FJAE on a standard dataset. We estimate for all examples their robustness w.r.t. to the inner-sampling and the outer-sampling uncertainty sets.

5.1. **Experimental Setup.**

For our experiments, we used TensorFlow [1] to train networks for recognizing images of the Fashion-MNIST dataset [23], which contains a training set of 60,000 fashion pictograms with corresponding labels of ten categories and a test set of 10,000 images of the same kind. The input domain is taken as $\mathscr{X} = [0, 255]^{784}$ and represents the possible pictures (we ignore integrality constraints on pixel values to avoid further integrality constraints in the MILP-models). For determining the outputs of $T^\star$ we used our own perception. For example, for all given adversarial examples, we checked, if one would reasonably include them into the same class as the respective original image.

For a fixed nominal image of each of the ten classes of apparel we did the following: We built 5 different DNNs with the same DNN-structure. Each DNN had an input layer of 784 neurons, representing the 28×28 pixels of the input image, 3 inner layers with 8 neurons each and an

output layer of 10 neurons, one for each possible class. All neurons of the inner layers were given as rectified linear units (ReLUs).

We independently trained our DNNs using stochastic gradient descent and cross-entropy loss for 50 epochs, which finished in minutes for each DNN. Our models achieved an accuracy between 85.76% and 87.39% on the training data. The accuracy on the test data is between 83.22% and 85.12%. For comparison: The human performance on the Fashion-MNIST dataset is estimated to be at 83.5% [23]. This is a surprisingly poor performance that can be attributed to the low resolution of the images. Unfortunately, for benchmarks with higher resolution the computational effort would have been prohibitive.

All MILPs have been solved using the publically available software PySCIPOpt [18, 12]. All norms presented are L_1-norms. The ε used to define the uncertainty sets was $\varepsilon = 0.05$ for the inner-sampling uncertainty set and $\varepsilon = 0.2$ for the outer-sampling uncertainty set. Note that those values are unrelated and were chosen by the requirement that the DNNs in the uncertainty sets should have similar classification-functions with a similar accuracy. A larger ε of the inner-sampling uncertainty set would have reduced the accuracy of the DNN on the test-data by too much. The value of 0.05 corresponds to a mild retraining on only one additional training example (see below). A smaller ε of the the outer-sampling uncertainty set would have led to too many rejected samples. The numbers of robustification samples to represent the uncertainty sets together with the nominal DNN in the optimization models was set to $\alpha = 5$. This balances the wish to have many samples with the requirement to be able to solve the resulting optimization problems fast enough. The cross-evaluation of all examples (FJRE, IRAE, ORAE) w.r.t. the inner-sampling and outer-sampling uncertainty sets was carried out on 15 fresh test-samples in the respective uncertainty sets.

As results, we compiled the probabilities of the DNNs to return the true class and the target class, the average classification probabilities, and the sum of probabilities for all other classes with non-maximal classification probabilities. Furthermore, we report the *average weak confidence levels*, which we define as the difference in classification probabilities between the target class and the true class. Note, we consider misclassifications to other classes than the target class as uncontrolled failures that cannot be attributed to our method.

5.2. **Experimental Examples.** For the generation of the following examples, we consider a trained network $\bar{N}$ and a nominal input for each of the ten possible classes of apparel in the data. For example, for a fixed image of a pullover from the test data, the adversarial example shown in figure 1 can be obtained by feeding this image into the model of [13]. Taking $\alpha = 5$ samples of $\mathscr{Z}_\alpha^{\text{in}}$ and solving our model 4.1 for the same image, we obtain the inner-sample robust adversarial example shown in figure 2. Note, that inner-sample robustness can be interpreted as robustness to slight retraining of the same DNN. To demonstrate this, we retrained our 5 DNNs by refitting them to just one of the images of the training set for one additional epoch. The implications of this retraining for the accuracy on the test dataset and the inner distance between original and the retrained DNN are shown in table 1. This shows that only mild retraining results in DNNs that are close in both $\mathscr{N}^{\text{in}}$ and $\mathscr{N}^{\text{osd}}$. For the calculation of an outer-sampling robust adversarial example, we created a sampled neighborhood $\mathscr{Z}_5^{\text{osd}}$ of $\bar{N}$ consisting of the networks $\bar{N}, \tilde{N}_1, \ldots, \tilde{N}_5$. For the approximate outer distance d^{osd}, we used the first 50 images contained in the training set as $\bar{X}$. Information on the DNNs $\tilde{N}_1, \ldots, \tilde{N}_5$ is given in table 2. This shows that DNNs that are close in $\mathscr{N}^{\text{osd}}$ can be very far apart in $\mathscr{N}^{\text{in}}$.

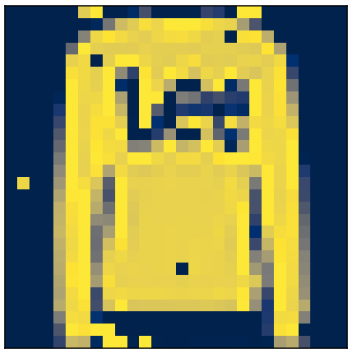


FIGURE 1. FJAE: An adversarial example for the class "pullover" calculated with FJ-model in [13] and $\gamma = 0.2$

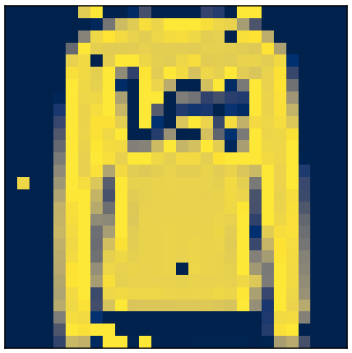


FIGURE 2. IRAE: An inner-sampling robust adversarial example for the class "pullover" with parameters $\gamma = 0.2$, $\varepsilon = 0.05$ and $\alpha = 5$

retraining no.	accuracy before	accuracy after	d^{in}	d^{osd}
1	84.84%	84.81%	0.03079	0.00198
2	85.12%	85.13%	0.05666	0.00304
3	84.83%	84.86%	0.09492	0.00817
4	84.12%	84.10%	0.00873	0.00066
5	84.30%	84.32%	0.06660	0.00501

TABLE 1. Overview of inner-sampling distances and outer-sampling pseudodistances for five different models after small "retraining"

Table 3 shows that the increases in the objective functions (which determine the costs of robustness) are higher for the ORAEs, i.e., their total deviation from the original input is larger than the deviation for the IRAEs and the FJAEs. Moreover, we see that the solution times are acceptable though the weakness of the Big-M-constraints leads to large numbers of branch-and-bound nodes.

Table 4 shows the evaluation of all examples w.r.t. the inner-sampling neighborhood. It is striking that all AEs (including the nominal FJAE) are robust against perturbations of weights and biases this small ($\varepsilon = 0.05$). A plausible explanation for this is that the confidence level

DNN	accuracy on test data	$d^{\text{osd}}(\bar{N},\tilde{N})$	$d^{\text{in}}(\bar{N},\tilde{N})$
$\tilde{N}_1$	84.50%	0.14960	926.93145
$\tilde{N}_2$	84.74%	0.13601	970.53378
$\tilde{N}_3$	84.46%	0.14474	918.99309
$\tilde{N}_4$	84.95%	0.14590	949.78444
$\tilde{N}_5$	84.32%	0.10915	892.70951

TABLE 2. Overview of the sampled DNNs in the outer-sampling neighborhood of $\bar{N}$

true class → target class	AE	#vars (#bin, #con)	#const	opt. val	cpu time (s)	#B&B-nodes
2→1	FJAE	1772 (68, 1704)	2565	12.96978	4.26	245
	IRAE	6712 (408, 6304)	8334	12.97047	240.01	12365
	ORAE	6712 (408, 6304)	8334	17.95743	2583.26	192972
3→2	FJAE	1772 (68, 1704)	2565	11.66269	2.61	146
	IRAE	6712 (408, 6304)	8334	11.66349	287.11	27568
	ORAE	6712 (408, 6304)	8334	26.67617	724.69	51764
4→3	FJAE	1772 (68, 1704)	2565	10.84349	16.14	1368
	IRAE	6712 (408, 6304)	8334	10.84455	712.14	67016
	ORAE	6712 (408, 6304)	8334	13.52954	30948.71	925595
5→4	FJAE	1772 (68, 1704)	2565	7.13864	5.90	921
	IRAE	6712 (408, 6304)	8334	7.13903	178.91	20092
	ORAE	6712 (408, 6304)	8334	9.07107	26721.04	774492
6→5	FJAE	1772 (68, 1704)	2565	8.50062	5.41	378
	IRAE	6712 (408, 6304)	8334	8.50075	283.00	14265
	ORAE	6712 (408, 6304)	8334	19.95219	6992.87	377562
7→6	FJAE	1772 (68, 1704)	2565	10.49392	8.96	785
	IRAE	6712 (408, 6304)	8334	10.49442	1530.70	118516
	ORAE	6712 (408, 6304)	8334	15.34994	7634.02	626016
8→7	FJAE	1772 (68, 1704)	2565	9.88721	7.48	644
	IRAE	6712 (408, 6304)	8334	9.88818	287.75	11654
	ORAE	6712 (408, 6304)	8334	11.78623	4622.17	577305
9→8	FJAE	1772 (68, 1704)	2565	3.75070	9.32	1285
	IRAE	6712 (408, 6304)	8334	3.75102	544.40	29103
	ORAE	6712 (408, 6304)	8334	6.16432	1509.27	96774
0→9	FJAE	1772 (68, 1704)	2565	28.90331	16.12	963
	IRAE	6712 (408, 6304)	8334	28.90401	758.73	47348
	ORAE	6712 (408, 6304)	8334	43.52225	5420.18	325253
1→0	FJAE	1772 (68, 1704)	2565	6.87471	6.67	470
	IRAE	6712 (408, 6304)	8334	6.87596	230.79	17428
	ORAE	6712 (408, 6304)	8334	8.29834	312.67	16343

TABLE 3. Solution statistics of the three adversarial examples

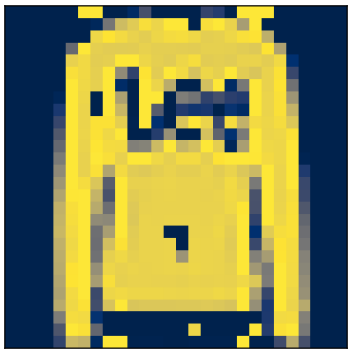


FIGURE 3. ORAE: An outer-sampling robust adversarial example for the class "pullover" with parameters $\gamma = 0.2$, $\varepsilon = 0.2$ and $\alpha = 5$

γ required in the FJ-model together with Lipschitz continuity w.r.t. weights and biases automatically generates a certain level of robustness of the FJAE. However, the outer-sampling robustness w.r.t. to DNNs trained from scratch is then much smaller for FJAE and IRAE compared to ORAE, as table 5 shows. That is, the ORAE is much more likely to stay an adversarial example for unknown DNNs in the outer-sampling uncertainty set for a test image in each of the ten classes. In particular, the superiority in outer-sampling robustness is not due to any special structure of a type of apparel in the experimental data. Our results indicate that the outer-sampling uncertainty set is most likely a disconnected set of many small inner-sampling uncertainty sets, and the IRAE has been computed to be robust against only one of them.

Finally, we tried a grey-box attack against a single given DNN. We chose this as the nominal DNN in the previous experimental results. This DNN can be regarded as a DNN run by a stakeholder who keeps the information on its weights and biased secret but uses a standard training method on public training data. The goal was to generate an AE that is an AE for the given DNN without knowing its weights and biases. For comparison, an FJAE and an IRAE were generated by training an artificial nominal DNN from scratch on the same training data with the same training method. Its weights and biases were then used to feed the FJ-model and the IRAE-model to generate an FJAE and an IRAE for the same fixed images in each class as in the previous experimental results. The ORAEs were generated by sampling the outer-sampling uncertainty set using the given DNN as an oracle to evaluate the outer-sampling distance to the given DNN. Then the ORAE-model was used to generate a single ORAE for the fixed image on each class. Afterwards, the classifications returned by the given DNN on all the FJAEs, the IRAEs and the ORAEs were compared.

The result of this preliminary gambling experiment was that for the given DNN to be attacked 1 out of 10 FJAEs were AEs, 0 out of 10 IRAEs were AEs, while 2 out of 10 ORAEs were AEs. Moreover, for the FJAEs the classification probability function of the DNN for the target class returned values very close to zero except for the one hit, which therefore could be named a "lucky punch". In contrast to this, for the ORAEs the classification probability function of the DNN for the target class returned much larger values, 4 times over 30%. See Figure 4 for detailed results in terms of a histogram for the number of instances with percentage values larger than or equal a given value. This histogram clearly indicates that the grey-box attack by the ORAE is far more powerful than the grey-box attacks by the FJAE or the IRAE.

true class → target class	AE	classification target	true	other miss	avg. activation level target	true	other miss	weak conf.
2→1	FJAE	100.00%	0.00%	0.00%	63.06%	2.00%	34.94%	38.82%
	IRAE	100.00%	0.00%	0.00%	63.07%	2.00%	34.93%	38.85%
	ORAE	100.00%	0.00%	0.00%	57.81%	2.69%	39.50%	34.02%
3→2	FJAE	100.00%	0.00%	0.00%	67.01%	12.75%	20.23%	54.26%
	IRAE	100.00%	0.00%	0.00%	67.02%	12.75%	20.23%	54.27%
	ORAE	100.00%	0.00%	0.00%	59.04%	6.80%	34.16%	41.82%
4→3	FJAE	100.00%	0.00%	0.00%	39.83%	16.37%	43.80%	23.46%
	IRAE	100.00%	0.00%	0.00%	39.84%	16.37%	43.79%	23.47%
	ORAE	100.00%	0.00%	0.00%	76.84%	17.11%	6.05%	59.73%
5→4	FJAE	100.00%	0.00%	0.00%	16.03%	14.69%	69.27%	1.34%
	IRAE	100.00%	0.00%	0.00%	16.04%	14.68%	69.28%	1.34%
	ORAE	100.00%	0.00%	0.00%	16.34%	15.31%	68.35%	1.02%
6→5	FJAE	100.00%	0.00%	0.00%	48.19%	24.45%	27.36%	23.74%
	IRAE	100.00%	0.00%	0.00%	48.20%	24.45%	27.35%	23.75%
	ORAE	100.00%	0.00%	0.00%	40.79%	27.96%	31.24%	12.83%
7→6	FJAE	100.00%	0.00%	0.00%	36.75%	0.00%	63.25%	19.08%
	IRAE	100.00%	0.00%	0.00%	36.77%	0.00%	63.23%	19.09%
	ORAE	100.00%	0.00%	0.00%	46.51%	0.00%	53.49%	26.31%
8→7	FJAE	100.00%	0.00%	0.00%	72.36%	21.75%	5.90%	50.61%
	IRAE	100.00%	0.00%	0.00%	72.39%	21.72%	5.89%	50.67%
	ORAE	100.00%	0.00%	0.00%	78.67%	20.25%	1.08%	58.41%
9→8	FJAE	100.00%	0.00%	0.00%	43.73%	18.93%	37.34%	24.80%
	IRAE	100.00%	0.00%	0.00%	43.76%	18.93%	37.31%	24.83%
	ORAE	100.00%	0.00%	0.00%	73.70%	25.76%	0.54%	47.94%
0→9	FJAE	100.00%	0.00%	0.00%	47.12%	17.40%	35.48%	29.72%
	IRAE	100.00%	0.00%	0.00%	47.15%	17.39%	35.46%	29.76%
	ORAE	100.00%	0.00%	0.00%	36.57%	18.17%	45.26%	18.39%
1→0	FJAE	100.00%	0.00%	0.00%	78.25%	13.41%	8.34%	64.84%
	IRAE	100.00%	0.00%	0.00%	78.28%	13.38%	8.34%	64.89%
	ORAE	100.00%	0.00%	0.00%	70.99%	11.61%	17.40%	59.37%

TABLE 4. Classification of the three examples by 15 sample-DNNs of the inner-sampling neighborhood

Though this increase in the final hit rate of the target class in this one-shot experiment is not overwhelming, one has to keep in mind that we have undertaken only three single grey-box attacks of a single DNN with all the effects of luck mixing with the methodological effects. This, of course, cannot replace a "clinical" study assessing the effectivity of either method on the basis of statistical evidence.

6. CONCLUSIONS

We have seen, that sampling can be a useful method to approach an adjustable robust linear program that is otherwise intractable. Just the small number of five additional DNN samples

true class → target class	AE	classification target	true	other miss	avg. activation level target	true	other miss	weak conf.
2→1	FJAE	6.67%	86.67%	6.67%	8.51%	81.77%	9.72%	-75.92%
	IRAE	6.67%	86.67%	6.67%	8.51%	81.77%	9.72%	-75.91%
	ORAE	66.67%	26.67%	6.67%	63.88%	25.99%	10.13%	33.53%
3→2	FJAE	0.00%	86.67%	13.33%	1.13%	77.46%	21.41%	-80.14%
	IRAE	0.00%	86.67%	13.33%	1.13%	77.46%	21.42%	-80.14%
	ORAE	26.67%	20.00%	53.33%	28.83%	17.13%	54.04%	-17.51%
4→3	FJAE	6.67%	13.33%	80.00%	8.67%	16.69%	74.64%	-35.62%
	IRAE	6.67%	13.33%	80.00%	8.67%	16.69%	74.64%	-35.62%
	ORAE	60.00%	26.67%	13.33%	45.05%	31.61%	23.33%	10.22%
5→4	FJAE	6.67%	40.00%	53.33%	6.61%	34.76%	58.63%	-46.34%
	IRAE	6.67%	40.00%	53.33%	6.61%	34.76%	58.63%	-46.33%
	ORAE	26.67%	20.00%	53.33%	23.76%	17.63%	58.61%	-21.52%
6→5	FJAE	6.67%	86.67%	6.67%	6.38%	73.21%	20.41%	-71.48%
	IRAE	6.67%	86.67%	6.67%	6.38%	73.21%	20.41%	-71.48%
	ORAE	53.33%	6.67%	40.00%	45.01%	9.65%	45.34%	7.23%
7→6	FJAE	6.67%	6.67%	86.67%	10.11%	6.67%	83.22%	-57.47%
	IRAE	6.67%	6.67%	86.67%	10.11%	6.67%	83.22%	-57.46%
	ORAE	33.33%	0.00%	66.67%	29.42%	0.02%	70.56%	-18.19%
8→7	FJAE	13.33%	66.67%	20.00%	10.37%	66.16%	23.47%	-72.20%
	IRAE	13.33%	66.67%	20.00%	10.37%	66.16%	23.47%	-72.20%
	ORAE	33.33%	60.00%	6.67%	35.14%	51.95%	12.90%	-23.31%
9→8	FJAE	0.00%	66.67%	33.33%	0.96%	56.69%	42.35%	-79.05%
	IRAE	0.00%	66.67%	33.33%	0.96%	56.70%	42.34%	-79.06%
	ORAE	66.67%	20.00%	13.33%	54.31%	22.22%	23.47%	15.71%
0→9	FJAE	0.00%	20.00%	80.00%	0.00%	29.73%	70.27%	-67.94%
	IRAE	0.00%	20.00%	80.00%	0.00%	29.73%	70.27%	-67.94%
	ORAE	6.67%	26.67%	66.67%	6.76%	20.87%	72.37%	-50.39%
1→0	FJAE	13.33%	86.67%	0.00%	12.94%	76.40%	10.67%	-64.09%
	IRAE	13.33%	86.67%	0.00%	12.94%	76.39%	10.67%	-64.08%
	ORAE	46.67%	40.00%	13.33%	35.03%	35.84%	29.13%	-15.44%

TABLE 5. Classification of the 3 examples by 15 sample-DNNs of the outer-sampling neighborhood

could lead to a noticeably improved outer-sampling robustness of the ORAE. Furthermore, outer-sampling robustness has shown to be the more meaningful of our two variants. On the one hand, the ORAE achieves almost the same level of inner-sampling robustness, while the IRAE could not show any better outer-sampling robustness than the FJAE. On the other hand, outer-sampling robustness comes with the advantage that one can simply exclude the nominal DNN from the model whenever one cannot rely on information on the weights and biases of it.

However, outer-sampling robustness comes with a cost. The calculation of an ORAE with standard MILP-methods ist harder than the calculation of an IRAE, even for the same number of networks to be included in the uncertainty sets. Furthermore, the ORAE deviates further

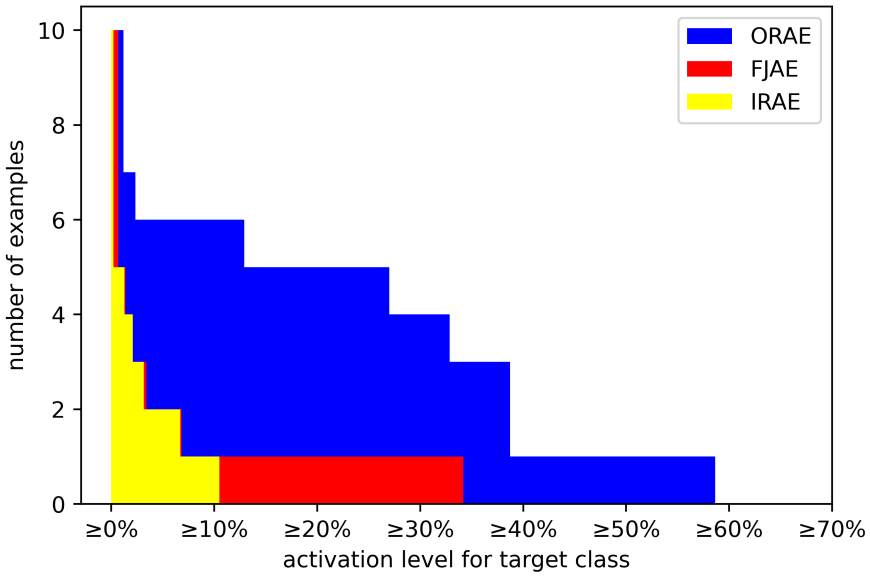


FIGURE 4. A histogram of the numbers of instances with classification probabilities at least a given value of the given DNN for the target class on the FJAE/IRAE/ORAE grey-box attacks.

away from the original input than the inner robust example does. This again emphasizes, that the set of DNNs with similar input-output relations is larger than the set of DNNs with similar internal parameters.

Clearly, this paper could only give preliminary evidence about benefits of the proposed method. In order to apply the method to more general settings, future research is needed. One direction concerns the representation of DNNs as MILPs. This could speed-up the necessary calculations and could, therefore, allow to handle more advanced DNN-structures. Such DNN-structures encompass DNN-structures with more layers and neurons, in particular more input neurons (larger resolution and bit-depths in the case of images) or convolutional networks (taking into account spatial information of the pixels in an image). Since this paper strongly relies on the FJ-model in [13], any improvement for the FJ-model would immediately lead to computational improvements for our approach. Sampling has shown to be a useful method for approaching our adjustable robust MILP. Thus, research on finding "good" samples for adjustable robust MILP would immediately be effective in our method. It would, therefore, be interesting to identify particularly representative samples in the outer ε-neighborhoods.

Finally, the further investigation of grey-box attacks to DNNs with unknown internal parameters need a careful experimental setup and well-founded statistical analysis to achieve conclusive results.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit

Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Naveed Akhtar, Ajmal Mian, Navid Kardan, and Mubarak Shah. Advances in adversarial attacks and defenses in computer vision: A survey. *IEEE Access*, 9:155161–155196, 2021.

[3] Tatsuya Akutsu and Hiroshi Nagamochi. A mixed integer linear programming formulation to artificial neural networks. In *Proceedings of the 2nd international conference on information science and systems*, pages 215–220, 2019.

[4] Ross Anderson, Joey Huchette, Will Ma, Christian Tjandraatmadja, and Juan Pablo Vielma. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1):3–39, 2020.

[5] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *International conference on machine learning*, pages 284–293. PMLR, 2018.

[6] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, Princeton, 2009.

[7] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical programming*, 99(2):351–376, 2004.

[8] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[9] Dimitris Bertsimas and Constantine Caramanis. Adaptability via sampling. In *2007 46th IEEE Conference on Decision and Control*, pages 4717–4722, 2007.

[10] Dimitris Bertsimas and Iain Dunning. Multistage robust mixed-integer optimization with adaptive partitions. *Operations Research*, 64(4):980–998, 2016.

[11] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer, 1997.

[12] Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, Christoph Graczyk, Katrin Halbig, Ivo Hedtke, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Dominik Kamp, Thorsten Koch, Kevin Kofler, Jurgen Lentz, Julian Manns, Gioni Mexi, Erik Mühmer, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Mark Turner, Stefan Vigerske, Dieter Weninger, and Lixing Xu. The SCIP Optimization Suite 9.0. Technical report, Optimization Online, February 2024.

[13] Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23, 07 2018.

[14] Claudio Gambella, Bissan Ghaddar, and Joe Naoum-Sawaya. Optimization problems for machine learning: A survey. *European Journal of Operational Research*, 290(3):807–828, 2021.

[15] Marc Goerigk and Anita Schöbel. Algorithm engineering in robust optimization. In *Algorithm engineering: selected results and surveys*, pages 245–279. Springer, 2016.

[16] Jindong Gu, Xiaojun Jia, Pau de Jorge, Wenqian Yu, Xinwei Liu, Avery Ma, Yuan Xun, Anjun Hu, Ashkan Khakzar, Zhijiang Li, Xiaochun Cao, and Philip Torr. A survey on transferability of adversarial examples across deep neural networks. *Transactions on Machine Learning Research*, 2024.

[17] Martin Gubri, Maxime Cordy, Mike Papadakis, Yves Le Traon, and Koushik Sen. Lgv: Boosting adversarial example transferability from large geometric vicinity. In *European Conference on Computer Vision*, pages 603–618. Springer, 2022.

[18] Stephen Maher, Matthias Miltenberger, João Pedro Pedroso, Daniel Rehfeldt, Robert Schwarz, and Felipe Serrano. PySCIPOpt: Mathematical programming in python with the SCIP optimization suite. In *Mathematical Software – ICMS 2016*, pages 301–307. Springer International Publishing, 2016.

[19] Krzysztof Postek and Dick den Hertog. Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS Journal on Computing*, 28(3):553–574, 2016.

[20] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[21] Vincent Tjeng, Kai Y. Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations*, 2019.

[22] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

[23] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[24] İhsan Yanıkoğlu, Bram L. Gorissen, and Dick den Hertog. A survey of adjustable robust optimization. *European Journal of Operational Research*, 277(3):799–813, 2019.