Overcoming the Optimizer's Curse: Obtaining Realistic Prescriptions from Neural Networks

Asterios Tsiourvas [1] **Georgia Perakis** [1]

Abstract

We study the problem of obtaining optimal and realistic prescriptions when using neural networks for data-driven decision-making. In this setting, the network is used to predict a quantity of interest and then is optimized to retrieve the decisions that maximize the quantity (e.g. find the best prices that maximize revenue). However, optimizing over-parameterized models often produces unrealistic prescriptions, far from the data manifold. This phenomenon is known as the Optimizer's Curse. To tackle this problem, we model the requirement for the resulting decisions to align with the data manifold as a tractable optimization constraint. This is achieved by reformulating the highly non-linear Local Outlier Factor (LOF) metric as a single linear or quadratic constraint. To solve the problem efficiently for large networks, we propose an adaptive sampling algorithm that reduces the initial hard-to-solve optimization problem into a small number of significantly easier-to-solve problems by restricting the decision space to realistic polytopes, i.e. polytopes of the decision space that contain at least one realistic data point. Experiments on publicly available networks demonstrate the efficacy and scalability of our approach.

1. Introduction

In recent years, deep learning has proven to be an extremely powerful tool for solving a wide variety of problems in various fields such as computer vision, natural language understanding, and biology (LeCun et al., 2015; Jumper et al., 2021; Min et al., 2023). Even though most applications have focused on prediction and generation, optimizing trained neural networks for decision-making is a question that has recently emerged.

Practitioners and academics have been increasingly incorporating machine learning in data-driven optimization problems for decision-making. By leveraging the predictive capabilities of deep learning, it is possible to model complex systems more accurately and, as a result, suggest more informed decisions. There are multiple areas where this paradigm is followed in practice. In operations management (Carbonneau et al., 2008; Glaeser et al., 2019; Ban & Rudin, 2019), machine learning models can be used to predict and then optimize the allocation of resources. In revenue management (Ferreira et al., 2016; Ettl et al., 2020; Chen et al., 2022) machine learning models can be trained on data including historical sales, market trends, and customer demographics to predict future revenue and then optimize to obtain informed decisions about how to set prices that maximize revenue. In healthcare, (Bertsimas et al., 2016; Fairley et al., 2019; Bertsimas et al., 2020), machine learning models can be used to predict the outcomes of clinical trials to find the optimal combination of regimens to be tested.

As a general rule, the more accurate the model is at predicting the outcome of interest, the better the model will be able to inform decision-making processes. Typically, highly accurate models, such as neural networks, are highly complex. However, when using a complex model to predict a quantity of interest and then optimize over it, the resulting prescription may not be representative of the actual data. The reason for this behavior is that in many real-world applications, data lie on or near manifolds characterized by significantly lower dimensions than the corresponding input space (Lin & Zha, 2008; Osher et al., 2017). In deep learning, although there is evidence (Basri & Jacobs, 2016) indicating that neural networks can effectively extract the intrinsic, low-dimensional coordinates of data, optimizing an over-parameterized neural network may yield unrealistic prescriptions, i.e. solutions in the input space that have low or not at all data density. This phenomenon extends to general highly complex machine learning models and is known as the "Optimizer's Curse" (Smith & Winkler, 2006).

In this work, we study the problem of generating both *optimal* and *realistic* prescriptions when optimizing over trained

[1]Operations Research Center, Massachusetts Institute of Technology, USA. Correspondence to: Asterios Tsiourvas <atsiour@mit.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

neural networks for data-driven decision-making. We first focus on neural networks with ReLU activations, which are among the most widely used in deep learning (Agarap, 2018), and then we extend our method to general neural networks. In particular, we focus on the optimization problem:

$$\begin{aligned} \max_{\boldsymbol{x}\in\mathcal{P}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x}\ is\ realistic, \end{aligned} \tag{1}$$

where the decision space is a bounded polyhedron $\mathcal{P}$ and we maximize a quantity predicted by a neural network $f(\boldsymbol{x})$ under the constraint that the resulting decision should be realistic. The main contributions of the paper are:

- We show how to explicitly model the requirement that the resulting decision should align with the data manifold as a tractable optimization constraint. This is achieved by reformulating the highly non-linear LOF (Breunig et al., 2000), a widely used metric for determining whether a new data point is an inlier or outlier with respect to the existing data, as a single linear or quadratic constraint.
- We propose a tractable, adaptive, sampling-based algorithm for obtaining realistic prescriptions from ReLU networks. The algorithm takes advantage of the geometry of ReLU networks and reduces the initial hard-to-solve optimization problem (mixed-integer program) into a small number of significantly easier-to-solve problems (linear or second-order cone programs). This is achieved by restricting the search space to realistic polytopes, i.e. polytopes of the decision space generated by the network that contain at least one realistic data point, and carefully selecting a subset of realistic polytopes as the search space. We then extend the algorithm to any differentiable model or any model that can be expressed using mixed-integer constraints, including linear regression and tree ensembles.
- We empirically evaluate our proposed method by conducting numerous experiments on publicly available, pre-trained neural networks. The results demonstrate that our method consistently yields realistic decisions with the highest objective values compared to other state-of-the-art optimization strategies and can scale for neural networks with millions of parameters. Additionally, we showcase the effectiveness of the proposed realistic constraint by visualizing the solution of the optimization algorithm with and without the proposed constraint on real-world datasets.

2. Related Literature

A crucial requirement for data-driven decision-making is realism in the decisions. Given that complex models suffer from the Optimizer's Curse, decision-makers have focused on simpler models, like linear regression and shallow decision trees, to ensure realism (Bertsimas et al., 2016; Ferreira et al., 2016; Cohen et al., 2020). However, such approaches do not utilize the predictive capabilities of deep learning and impose realism in the decisions implicitly through low model complexity. In contrast, we show how neural networks can be optimized to obtain realistic prescriptions by *explicitly* imposing realism by reformulating the highly nonlinear LOF (Breunig et al., 2000), one of the most widely used metrics for verifying whether a data point is an inlier, as a single tractable constraint in the optimization problem. LOF has been mostly used in the field of counterfactual explanations (Guidotti, 2022; Dutta et al., 2022; Lucic et al., 2022). Except for Kanamori et al. (2020), which utilized LOF as a regularization term for generating counterfactual explanations for linear and tree models, and Tsiourvas et al. (2024) which used it as an explicit constraint to generate counterfactual explanations solely for ReLU networks, existing works have used LOF as an evaluation metric.

In recent years, deep learning has gained significant attention due to its exceptional performance on a wide variety of tasks (LeCun et al., 2015; Jumper et al., 2021; Min et al., 2023). Among deep learning architectures, ReLU networks have gained significant attention because of their piecewise linear nature (Lee et al., 2019) that allows for analytical tractability. Fischetti & Jo (2018) first showed that optimizing over an already trained ReLU network can be expressed as a mixed-integer optimization program. Since then, multiple works have focused on optimizing already trained ReLU networks, utilizing both mixed-integer optimization (Anderson et al., 2020; De Palma et al., 2021; Tsay et al., 2021) and approximate methods (Katz et al., 2017; 2019; Perakis & Tsiourvas, 2022; Tong et al., 2024). This structure has also been utilized for a variety of applications, such as robustness verification (Tjeng et al., 2017), network compression (Serra et al., 2020), prescriptive analytics (Sun & Tsiourvas, 2023) and molecular design (McDonald et al., 2023).

To optimize ReLU networks, most researchers have focused on MIP-based algorithms (Huchette et al., 2023). Even though such approaches offer optimality guarantees on the solution, their scalability is stressed when the network has more than $2-3$ hidden layers (Strong et al., 2023). On the other hand, to optimize general deep learning models, most researchers have focused on gradient-based methods (Kingma & Ba, 2014; Loshchilov & Hutter, 2017) due to their scalability, failing however to provide guarantees regarding the optimality and the realism of the solution. In this work, we tackle these problems by introducing a general, scalable adaptive sampling algorithm that reduces the initial hard-to-solve optimization problem into a small number of easier-to-solve problems and leads to *certified realistic solutions* with high objective value. In what follows we

introduce ReLU networks and their underlying geometry, which we then utilize to reformulate LOF into a single tractable constraint.

3. Methodology

3.1. ReLU Neural Networks

We begin with some useful definitions. Let $\mathcal{P} \subseteq \mathbb{R}^d$ denote the input space that we assume to be a bounded polyhedron and let $\mathcal{D} = \{x_i \in \mathcal{P}\}_{i=1}^m$ be the dataset on which the network was trained. Throughout the paper, the terms decision space, feasible region, and domain of the input will be used interchangeably. We also define that $[n] := \{1, \ldots, n\}$. We denote the ReLU network as the function $f_\theta : \mathcal{P} \to \mathcal{Y} \subseteq \mathbb{R}$, where θ is the set of weights of the network and $\mathcal{Y}$ is the output space. The analytical form of f_θ is

$$f_\theta(\boldsymbol{x}) = \boldsymbol{w}_L^T \boldsymbol{x}_{L-1} + b_L, \boldsymbol{x}_l = \sigma(\boldsymbol{W}_l \boldsymbol{x}_{l-1} + \boldsymbol{b}_l), \forall l \in [L],$$

where L is the number of hidden layers, n_l is the number of neurons of the hidden layer l, $\sigma(\cdot) : \mathbb{R} \to \mathbb{R}$ is the non-linear, continuous and sub-differentiable ReLU activation function, i.e. $\sigma(x) = \max\{x, 0\}$, $\boldsymbol{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$ is the weight matrix of layer l, $\boldsymbol{b}_l \in \mathbb{R}^{n_l}$ is the bias term of layer l and $\theta := \{\boldsymbol{w}_L, b_L, \ldots, \boldsymbol{W}_1, \boldsymbol{b}_1\}$. For ease of notation, we define $\boldsymbol{x}_0$ to be equal to the input vector $\boldsymbol{x} \in \mathcal{P}$ with $n_0 := d$. Therefore, problem (1) can be re-written as

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}_0 \in \mathcal{P}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{L-1}} & \boldsymbol{w}_L^T \boldsymbol{x}_{L-1} + b_L \\ \text{subject to} & \boldsymbol{x}_l = \max\{\boldsymbol{W}_l \boldsymbol{x}_{l-1} + \boldsymbol{b}_l, 0\}, \ \forall l \in [L], \\ & \boldsymbol{x}_0 \ \textit{is realistic}. \end{array} \tag{2}$$

Due to the non-linear nature of the ReLU activation function, even without the realistic constraint, solving problem (2) directly is a computationally challenging task since problem (2) is a non-convex, non-linear optimization problem.

3.2. A Mixed-Integer Optimization Approach

Recent research has focused on solving problem (2) (without the realistic constraint) by reformulating the equality constraint using mixed-integer programming (MIP) techniques. Fischetti & Jo (2018) first proposed a re-formulation of the equality constraint into a set of mixed-integer optimization constraints. Specifically, for layer l, the authors observed that the constraint $\boldsymbol{x}_l = \max\{\boldsymbol{W}_l \boldsymbol{x}_{l-1} + \boldsymbol{b}_l, 0\}$ is equivalent to the constraint $\boldsymbol{x}_l \in \mathcal{C}(\boldsymbol{x}_{l-1})$ where

$$\mathcal{C}(\boldsymbol{x}_{l-1}) = \left\{ \boldsymbol{y} \;\middle|\; \begin{array}{l} \boldsymbol{y} \geq \boldsymbol{W}_l \boldsymbol{x}_{l-1} + \boldsymbol{b}_l, \\ \boldsymbol{y} \leq \boldsymbol{W}_l \boldsymbol{x}_{l-1} + \boldsymbol{b}_l - \boldsymbol{l}_l \odot (1 - \boldsymbol{z}_l), \\ \boldsymbol{y} \leq \boldsymbol{u}_l \odot \boldsymbol{z}_l, \\ \boldsymbol{y} \geq 0, \boldsymbol{z}_l \in \{0,1\}^{n_l} \end{array} \right\}.$$

In this definition, n_l is the number of neurons at hidden layer l, $\boldsymbol{z}_l \in \{0,1\}^{n_l}$ is a vector of binary variables that indicates whether a neuron of layer l is activated or not, $\boldsymbol{u}_l \in \mathbb{R}_l^n$ is a vector of upper bound values for the output of each neuron of layer l, $\boldsymbol{l}_l \in \mathbb{R}^{n_l}$ is a vector of lower bounds for the output of each neuron of layer l and $\odot$ denotes the element-wise product between two vectors. Fischetti & Jo (2018) suggested that $\boldsymbol{u}_l = M \cdot \boldsymbol{1}, \boldsymbol{l}_l = -M \cdot \boldsymbol{1}$ where $M > 0$ is a large positive value and $\boldsymbol{1}$ is a vector of dimension n_l that contains 1 at each position. For $\boldsymbol{x}_0$, the upper and lower bounds are obtained from $\mathcal{P}$. By combining all of the above, we obtain the following MIP re-formulation of problem (2).

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}_0 \in \mathcal{P}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{L-1}, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_L} & \boldsymbol{w}_L^T \boldsymbol{x}_{L-1} + b_L \\ \text{subject to} & \boldsymbol{x}_l \in \mathcal{C}(\boldsymbol{x}_{l-1}), \ \forall l \in [L], \\ & \boldsymbol{x}_0 \ \textit{is realistic}. \end{array} \tag{3}$$

3.3. Geometry of ReLU Neural Networks

The previous reformulation shows analytically that ReLU networks are piecewise linear functions. Given a trained ReLU neural network and a fixed activation pattern, i.e. a fixed configuration of the activation values or equivalently of the binary variables $\boldsymbol{z}$ of problem (3), the ReLU network reduces into a linear model (Huchette et al., 2023) over the feasible set defined by the activation pattern. For a given activation pattern, this feasible set is a polyhedron that is a subset of the input space $\mathcal{P}$ (Sun & Tsiourvas, 2023). Feasible sets, coming from all feasible activation patterns, partition $\mathcal{P}$ into a finite number of polyhedra such that $\mathcal{P}_j \cap \mathcal{P}_{j'} = \emptyset, \ \forall j \neq j'$, and $\cup_j \mathcal{P}_j = \mathcal{P}$. To illustrate the partitioning scheme, we present an example in Figure 1.

Example 1: Let $\mathcal{P} = [0,1]^2$, and f be the trained ReLU network with one hidden layer shown in Figure 1 (left). We have $f(x) = -0.5 \max\{x_1 - x_2, 0\} + 2\max\{x_1 + x_2 - 0.5, 0\}$. By enumerating all possible activation patterns for the hidden layer, i.e. all possible combinations of ReLU activations for the two hidden neurons, we obtain four polytopes, i.e., $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ and $\mathcal{P}_4$, that partition the input space $\mathcal{P}$ as shown in Figure 1 (right). The partitions of the input space $\mathcal{P}$ and the corresponding linear model resulting from the ReLU network at each partition are:

- $\mathcal{P}_1 = \{(x_1, x_2) \in \mathcal{P} : x_1 - x_2 \geq 0, x_1 + x_2 - 0.5 \geq 0\}$, $f(x) = 1.5x_1 + 2.5x_2 - 1, \forall x \in \mathcal{P}_1$.
- $\mathcal{P}_2 = \{(x_1, x_2) \in \mathcal{P} : x_1 - x_2 \geq 0, x_1 + x_2 - 0.5 < 0\}$, $f(x) = -0.5x_1 + 0.5x_2, \forall x \in \mathcal{P}_2$.
- $\mathcal{P}_3 = \{(x_1, x_2) \in \mathcal{P} : x_1 - x_2 < 0, x_1 + x_2 - 0.5 < 0\}$, $f(x) = 0, \forall x \in \mathcal{P}_3$.
- $\mathcal{P}_4 = \{(x_1, x_2) \in \mathcal{P} : x_1 - x_2 < 0, x_1 + x_2 - 0.5 \geq 0\}$, $f(x) = 2x_1 + 2x_2 - 1, \forall x \in \mathcal{P}_4$.

It is seen that problem (3) can be solved optimally, by solving a subproblem over each feasible polytope $\mathcal{P}_j$ and then,

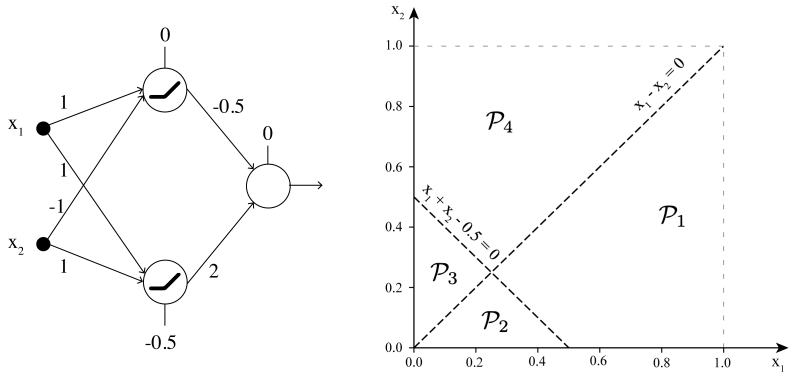

Figure 1. (Left) A ReLU network with one hidden layer. (Right) The partition of the decision space $\mathcal{P}$ by the network into 4 polytopes.

reporting the solution that leads to the highest value of f_θ. Each subproblem is significantly simplified since a fixed configuration of activations (equivalent to fixing all binary variables $\boldsymbol{z}$) results in the initially mixed-integer set of constraints $\boldsymbol{x}_l \in \mathcal{C}(\boldsymbol{x}_{l-1}), l \in [L]$, to become a linear set of constraints. Omitting the realistic constraint, for the network of Example 1 we only need to solve 4 simpler subproblems (four linear programs (LPs); one over each $\mathcal{P}_j$) instead of a single harder-to-solve MIP and output as the optimal solution, the solution that gives the highest objective out of the 4 solutions. For general ReLU networks, this approach generalizes and requires solving N subproblems, where N is the number of all feasible polytopes.

4. Realistic Prescriptions

4.1. On the Realistic Constraint

So far, we have not focused on the constraint $\boldsymbol{x}$ *is realistic*. To model the constraint tractably and identify whether a data point is an outlier or not we use the LOF (Breunig et al., 2000). LOF is a well-known metric in the literature that quantifies whether a sample follows the underlying data distribution. It identifies anomalous data points by measuring the local deviation of a data point with respect to its neighbors in $\mathcal{D}$. We present the formal definition of LOF.

Definition 4.1. (Local Outlier Factor (LOF); Breunig et al. (2000)) *For $\boldsymbol{x} \in \mathcal{D}$, let $N_k(\boldsymbol{x})$ to be its $k-$Nearest Neighbors in $\mathcal{D}$. The $k-$reachability distance rd_k of $\boldsymbol{x}$ with respect to $\boldsymbol{x}'$ is defined by $rd_k(\boldsymbol{x}, \boldsymbol{x}') = \max\{\delta(\boldsymbol{x}, \boldsymbol{x}'), d_k(\boldsymbol{x}')\}$, where $d_k(\boldsymbol{x}')$ is the distance δ between $\boldsymbol{x}'$ and its $k-$th nearest instance in $\mathcal{D}$. The $k-$local reachability density of $\boldsymbol{x}$ is defined by $lrd_k(\boldsymbol{x}) = |N_k(\boldsymbol{x})|(\sum_{\boldsymbol{x}' \in N_k(\boldsymbol{x})} rd_k(\boldsymbol{x}, \boldsymbol{x}'))^{-1}$. Then, the $k-$LOF of $\boldsymbol{x}$ on $\mathcal{D}$ is defined as*

$$LOF_{k,\mathcal{D}}(\boldsymbol{x}) = \frac{1}{|N_k(\boldsymbol{x})|} \sum_{\boldsymbol{x}' \in N_k(\boldsymbol{x})} \frac{lrd_k(\boldsymbol{x}')}{lrd_k(\boldsymbol{x})}. \quad (4)$$

For the distance metric $\delta : \mathcal{P} \times \mathcal{P} \to \mathbb{R}_{\geq 0}$, typically the ℓ_p norm with $p \in \{1, 2, \infty\}$ is used. In this paper, we focus on the case with $p = 2$. A value of $LOF_{k,\mathcal{D}}(\boldsymbol{x}) \lesssim 1$ indicates that $\boldsymbol{x}$ is an inlier, while $LOF_{k,\mathcal{D}}(x) \gg 1$ indicates that x is an outlier. In practice, to quantify whether $\boldsymbol{x}$ is an inlier or not, we check whether $LOF_{k,\mathcal{D}}(\boldsymbol{x})$ is less or equal to a predefined threshold t, that is close to 1. In section D of the Appendix, we argue on the choice of the LOF to model the realistic constraint, and we compare it with the Wasserstein distance that is widely used in the literature.

Example 2: In Figure 2 (left) we observe the partition of the input space $\mathcal{P}$ by the hidden layer of the ReLU network of the previous example along with the data points on which it was trained. In Figure 2 (right), we observe the LOF scores determined by the LOF classifier of scikit-learn (Buitinck et al. (2013)) for each data point. The LOF score can effectively identify both inliers and outliers for this example. For the data points in $\mathcal{P}_1, \mathcal{P}_2$ the LOF is ≈ 1, indicating that these data points are inliers. In contrast, for the isolated data point in $\mathcal{P}_4$ LOF is 8.65, a score significantly greater than 1, suggesting that this data point is an outlier.

To impose that the resulting prescription should be realistic, we incorporate the constraint $LOF_{k,\mathcal{D}}(\boldsymbol{x}) \leq t$ directly into problem (3). We obtain the following optimization problem:

$$\begin{aligned} \max_{\boldsymbol{x}_0 \in \mathcal{P}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{L-1}} \quad & \boldsymbol{w}_L^T \boldsymbol{x}_{L-1} + b_L \\ \text{subject to} \quad & \boldsymbol{x}_l \in \mathcal{C}(\boldsymbol{x}_{l-1}),\ \forall l \in [L], \\ & LOF_{k,\mathcal{D}}(\boldsymbol{x}_0) \leq t. \end{aligned} \quad (5)$$

Solving problem (5) to optimality leads to the realistic solution with the highest possible objective. However, there are 2 caveats. First, the constraint $LOF_{k,\mathcal{D}}(\boldsymbol{x}_0) \leq t$ in its current form is highly non-linear, and thus makes problem (5) intractable. Second, even if the constraint was tractable (for example, a linear constraint), solving problem (5) to optimality would still require solving an MIP, which would not scale for large ReLU networks, that are used in practice. In what follows, we take advantage of the underlying geom-

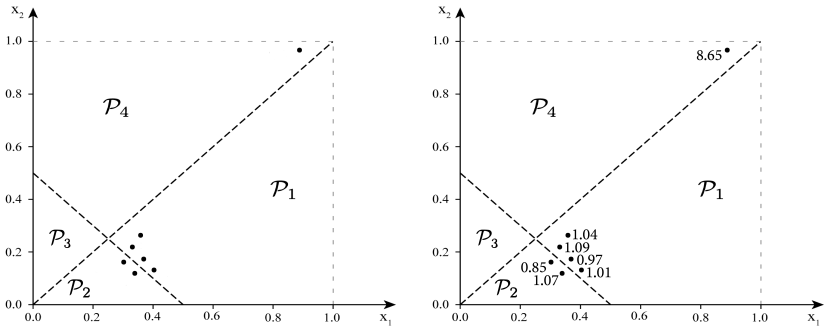

Figure 2. (Left) The partition of the input space $\mathcal{P}$ by the hidden layer of the ReLU network of Example 1 along with the dataset points on which the network was trained. (Right) The LOF score for each dataset point.

etry of ReLU networks and propose a tractable and scalable optimization algorithm for solving problem (5).

4.2. The Concept of Realistic Polytopes

LOF compares the local density of a data point to the local densities of its neighbors. A data point, to be considered realistic with respect to LOF, should be close to an existing point or group of data points that are inliers. This observation helps us reduce significantly the decision space to areas that contain realistic data points. To do so, we begin by introducing the concept of t-realistic polytopes.

Definition 4.2. (t-Realistic Polytope) *Given a trained ReLU network f_θ, a t-realistic polytope of f_θ is a feasible polytope generated by f_θ that contains at least one data point of the dataset with LOF score less or equal to t.*

Example 3: In Figure 2 (right) we observe that, according to the definition, $\mathcal{P}_1$ and $\mathcal{P}_2$ are 1.1-realistic polytopes as they contain at least one data point with LOF less than 1.1, while $\mathcal{P}_3$ and $\mathcal{P}_4$ are not.

In the paper, we will interchangeably use the terms t-realistic and realistic, as we assume that t will always be close to 1. By restricting the decision space to realistic polytopes, the problem's complexity decreases significantly. To find the optimal realistic solution we now need to solve at most $N_r \leq N$ subproblems, where N_r is the number of realistic polytopes. In the example, restricting the search space to realistic polytopes requires solving only 2 subproblems.

In the following theorem, we utilize the concept of t-realistic polytopes and the LOF score metric (with $k = 1$) to derive a tractable optimization formulation that leads to certified realistic prescriptions, while we also provide a tight upper bound on its optimal objective value.

Theorem 4.3. *Let f_θ be a ReLU network, $\mathcal{P}_j$ be a polytope of f_θ that contains at least one data point with $LOF_{1,\mathcal{D}} \leq t$, $\boldsymbol{z}^j$ be the values of $\mathcal{P}_j$'s corresponding activation pattern, $f_\theta(\cdot; \boldsymbol{z}^j)$ be the given ReLU network with fixed activation $\boldsymbol{z}^j$ and $\mathcal{D}_j := \mathcal{D} \cap \mathcal{P}_j$ be the set of points of $\mathcal{D}$ that belong to $\mathcal{P}_j$. We denote as $\boldsymbol{x}'$ the data point in $\mathcal{D}_j$ such that $LOF_{1,\mathcal{D}}(\boldsymbol{x}') \leq t$ and $f_\theta(\boldsymbol{x}') \geq f_\theta(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathcal{D}_j$. Then,*

1. ***(A Tractable Formulation)** The optimization problem*

$$\begin{aligned} \max_{\boldsymbol{x} \in \mathcal{P}} \quad & f_\theta(\boldsymbol{x}; \boldsymbol{z}^j) \\ \text{subject to} \quad & \delta(\boldsymbol{x}, \boldsymbol{x}') \leq t \cdot \max\{\delta(\boldsymbol{x}', \boldsymbol{x}''), d_1(\boldsymbol{x}''))\}, \\ & \boldsymbol{x}'' \in N_1(\boldsymbol{x}'), \end{aligned} \tag{6}$$

   retrieves the realistic solution $\boldsymbol{x}^* \in \mathcal{P}_j$ with the highest objective, $f_\theta(\boldsymbol{x}^*; \boldsymbol{z}^j) \geq f_\theta(\boldsymbol{x}'; \boldsymbol{z}^j)$.

2. ***(A Tight Upper Bound on the Objective)** The objective value of the optimal solution of problem (6) is upper bounded as $f_\theta(\boldsymbol{x}^*; \boldsymbol{z}^j) \leq (\boldsymbol{w}^j)^T(\boldsymbol{x}' + r\frac{\boldsymbol{w}^j}{||\boldsymbol{w}^j||_2})$, where $\boldsymbol{w}^j \in \mathbb{R}^d$ is the vector of the weights that define the hyperplane of f_θ over $\mathcal{P}_j$ and $r := t \cdot \max\{\delta(\boldsymbol{x}', \boldsymbol{x}''), d_1(\boldsymbol{x}''))\}$, $\boldsymbol{x}'' \in N_1(\boldsymbol{x}')$. The bound is tight when $\boldsymbol{x}^* = \boldsymbol{x}' + r\frac{\boldsymbol{w}^j}{||\boldsymbol{w}^j||_2}$.*

The proof can be found in section A of the Appendix. The first part of the theorem shows that by searching over a realistic polytope at a time, LOF can be reformulated into a *single tractable constraint* (either linear or quadratic), and thus, problem (6) becomes solvable in a tractable and scalable manner using commercial solvers. The second part of the theorem provides a tight upper bound on the objective value of problem (6). This bound will be later used to restrict further the decision space and carefully select a small subset of realistic polytopes to search over. In section C of the Appendix we demonstrate experimentally that for $t = 1$ using the bound of Theorem 4.3.2 significantly reduces the search space by 87.5% to 99.9% depending on the underlying neural network.

Example 4: Consider the realistic polytope of Figure 3, with $\boldsymbol{z}$ the vector of the values of its activation pattern and $\boldsymbol{w}$ the

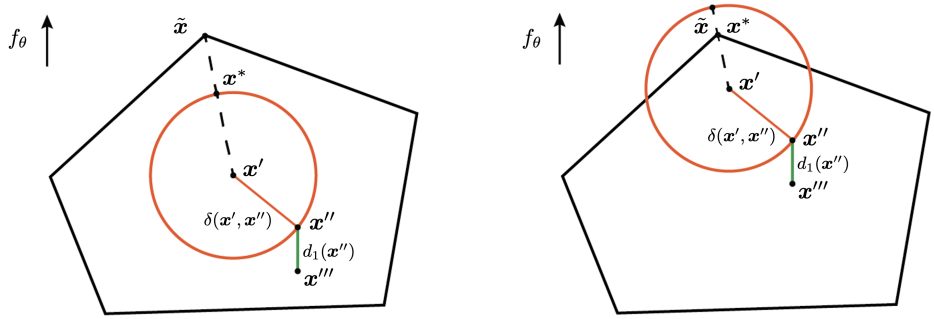

Figure 3. A realistic polytope of f_θ with optimal objective achieved at $\tilde{\boldsymbol{x}}$. (Left) The solution of problem (6), leads to an optimal solution $\boldsymbol{x}^*$ such that $f_\theta(\boldsymbol{x}^*) < f_\theta(\tilde{\boldsymbol{x}})$ and thus, the upper bound of Theorem 4.3 is tight. (Right) The solution of problem (6), leads to an optimal solution $\boldsymbol{x}^* = \tilde{\boldsymbol{x}}$. In this case, the upper bound of Theorem 4.3 is not tight.

vector of the weights that define the hyperplane of f_θ over this polytope. We denote as $\tilde{\boldsymbol{x}}$ the stationary point where f_θ retrieves its highest value within the realistic polytope. For the purpose of this example, we assume that $t = 1$. In Figure 3 (left), we observe that the solution of problem (6) leads to an optimal realistic solution $\boldsymbol{x}^*$ with a lower objective than $f_\theta(\tilde{\boldsymbol{x}})$. In this case, the upper bound is tight and $\boldsymbol{x}^* = \boldsymbol{x}' + r\frac{\boldsymbol{w}}{||\boldsymbol{w}||_2}$, where given that $d_1(\boldsymbol{x}'') < \delta(\boldsymbol{x}', \boldsymbol{x}'')$ and $t = 1$, r is equal to $t \cdot \max\{\delta(\boldsymbol{x}', \boldsymbol{x}''), d_1(\boldsymbol{x}'')) = \delta(\boldsymbol{x}', \boldsymbol{x}'')$. On the other hand, in Figure 3 (right), we observe that the solution $\boldsymbol{x}^*$ of problem (6) coincides with $\tilde{\boldsymbol{x}}$ and thus, produces the highest possible objective. In this case, we have that $f_\theta(\boldsymbol{x}^*; \boldsymbol{z}) < \boldsymbol{w}^T(\boldsymbol{x}' + r\frac{\boldsymbol{w}}{||\boldsymbol{w}||_2})$.

Remark 4.4. The maximum allowed movement $r = t \cdot \max\{\delta(\boldsymbol{x}', \boldsymbol{x}''), d_1(\boldsymbol{x}'')\}$, $\boldsymbol{x}'' \in N_1(\boldsymbol{x}')$, can be pre-computed on $\mathcal{D}$. Thus, if $p \in \{1, \infty\}$ problem (6) is an LP, and if $p = 2$ it is a second-order cone program (SOCP).

Remark 4.5. The realistic constraint of Theorem 4.3.1 applies to any model that can be expressed by linear mixed-integer constraints (e.g. linear regression, tree ensembles). We present the corresponding reformulations in section E of the Appendix.

As the size of f_θ and $\mathcal{D}$ increases, N_r may also increase, making the cost of searching over all realistic polytopes prohibitive. In what follows, we propose an adaptive sampling algorithm that carefully selects a small subset of realistic polytopes using the upper bound of Theorem 4.3.2.

4.3. An Adaptive Sampling Algorithm

The proposed algorithm consists of the following steps. First, the algorithm performs the feed-forward pass on every realistic data point of $\mathcal{D}$. By performing the feed-forward pass, the algorithm (1) retrieves the activation patterns that define each realistic polytope, (2) obtains the vector of the weights that define the hyperplane of f_θ over each polytope, (3) retrieves the inlier within each polytope with the highest objective and (4) calculates the upper bound of Theorem 4.3.2. By calculating the upper bound per realistic polytope, the proposed algorithm further reduces the search space, by eliminating realistic polytopes with upper bounds that are lower than the value of the inlier with the highest objective.

Algorithm 1 Sampling Algorithm for Realistic Prescriptions

Input: Training set $\mathcal{D}$, ReLU network f_θ, threshold t, number of samples n.
Initialize polytope dictionary $\mathcal{F} \leftarrow \{\}$, distribution dictionary $p \leftarrow \{\}$, $\boldsymbol{x}^* \leftarrow None$.
For all $\boldsymbol{x} \in \mathcal{D}$:
  if $LOF_{1,\mathcal{D}}(\boldsymbol{x}) \leq t$:
    Perform the feed-forward pass on $\boldsymbol{x}$ and retrieve activation pattern $\boldsymbol{z}$.
    Calculate weights $\boldsymbol{w}$ of hyperplane of f_θ for activation pattern $\boldsymbol{z}$, value $v = f_\theta(\boldsymbol{x})$ and upper bound u.
    if $\boldsymbol{z}$ not in $\mathcal{F}$ **or** $v > \mathcal{F}[\boldsymbol{z}][2]$:
      $\mathcal{F}[\boldsymbol{z}] \leftarrow (\boldsymbol{x}, \boldsymbol{w}, v, u)$
Remove from $\mathcal{F}$ all $\boldsymbol{z}$'s that have an upper bound u lower than an existing value v in $\mathcal{F}$.
Define for each $\boldsymbol{z} \in \mathcal{F}$ the sampling distribution $p[\boldsymbol{z}] \leftarrow softmax(u)$.
Draw n activation patterns from $\mathcal{F}$ according to p.
For all sampled $\boldsymbol{z}$, set $\boldsymbol{x}' \leftarrow \mathcal{F}[\boldsymbol{z}][0]$ and solve (6).
Return as $\boldsymbol{x}^*$ the solution with the highest value of f_θ.

After selecting the subset of realistic polytopes that have high enough upper bounds, we define the probability of optimizing over each remaining $\mathcal{P}_j$ as $p_j := \text{softmax}(u_j) = \frac{e^{u_j}}{\sum_i e^{u_i}}$, where u_j is the upper bound of $\mathcal{P}_j$ and the sum is taken over the subset of the remaining realistic polytopes $\mathcal{R}$. Finally, the algorithm samples $n \leq |\mathcal{R}|$ times without replacement from the distribution and solves problem (6) over each sampled realistic polytope. After n iterations, it returns as the optimal solution the inlier $\boldsymbol{x}^*$ that achieves the highest objective. Since for each sample, we solve a single LP or SOCP, this method takes advantage of the scalability

of the existing solvers and scales for large ReLU networks. The proposed method is presented in Algorithm 1.

We also present an upper bound on the number of samples required for the algorithm to retrieve the optimal solution.

Proposition 4.6. *The proposed sampling algorithm retrieves the optimal realistic solution using at most*

$$n \leq \min\left\{|D|, \sum_{(j_1,\ldots,j_L)\in J} \prod_{l=1}^{L} \binom{n_l}{j_l}\right\} \tag{7}$$

samples, where $J = \{(j_1,\ldots,j_L) \in \mathbb{Z}^L : 0 \leq j_L \leq \min\{n_0, n_1 - j_1, \ldots, n_{l-1} - j_{l-1}\}, \forall l = 1,\ldots,L\}$.

The proof can be found in section B of the Appendix. Intuitively, the proposition shows that for over-parameterized networks (common in practice), the maximum number of samples required to reach optimality is independent of the characteristics of the network (number of layers and neurons) and is equal to $|\mathcal{D}|$. In what follows, we utilize the previous results and extend our method to general differentiable machine learning models.

5. Extension to General Differentiable Models

We denote the general differentiable machine learning model as the function $f_\theta : \mathcal{P} \to \mathbb{R}$, where θ is the set of its parameters/weights. Let $\boldsymbol{x}'$ be a realistic data point in $\mathcal{D}$. Given that f_θ is differentiable we can perform projected gradient ascent starting from $\boldsymbol{x}'$ until convergence to retrieve a certified realistic solution with high objective. We utilize Theorem 4.3.1 and at each iteration k, the projected gradient ascent performs the step:

$$\boldsymbol{x}_{k+1} = Proj_{\mathcal{X}'}\left(\boldsymbol{x}_k + \alpha_k \nabla f_\theta(\boldsymbol{x}_k)\right), \tag{8}$$

where α_k is the step size, $\nabla f_\theta(\boldsymbol{x}_k)$ is the gradient of f_θ at $\boldsymbol{x}_k$ and $Proj_{\mathcal{X}'}$ is the projection operator onto the feasible set $\mathcal{X}' := \{\boldsymbol{x} \in \mathcal{P} : \delta(\boldsymbol{x}, \boldsymbol{x}') \leq t \cdot \max\{\delta(\boldsymbol{x}', \boldsymbol{x}''), d_1(\boldsymbol{x}'')\}\}$, $\boldsymbol{x}'' \in N_1(\boldsymbol{x}')\}$.

Given that the right-hand side of the constraint is precomputed, the realistic constraint depends solely on the distance function δ. In our experiments, we use the ℓ_2 norm and thus, the realistic constraint is a single convex quadratic constraint. By performing this procedure iteratively for different sampled $\boldsymbol{x}'$ from the distribution, as described in Algorithm 1, the method converges to the realistic solution with the highest objective. Given the efficacy and scalability of existing frameworks (Pytorch optimizers), this method can scale for complex deep learning models (Liu et al., 2023), as we show in section F of the Appendix.

We can also extend the upper bound of Theorem 4.3.2 under the assumption that f_θ is L-Lipschitz continuous. This is a valid assumption for deep learning models and there have been various works that propose tight upper bounds on L even for modern, transformer-based architectures (Kim et al., 2021). Following the proof of Theorem 4.3.2, we denote $\boldsymbol{x}_{ub} := \boldsymbol{x}' + r\frac{\nabla f_\theta(\boldsymbol{x}')}{||\nabla f_\theta(\boldsymbol{x}')||_2}$. Since f_θ is Lipschitz, we have that $||f_\theta(\boldsymbol{x}') - f_\theta(\boldsymbol{x}_{ub})||_2 \leq \hat{L} \cdot ||\boldsymbol{x}' - \boldsymbol{x}_{ub}||_2$, where $\hat{L} \geq L > 0$ is an estimate of L. Therefore, the upper bound is $f_\theta(\boldsymbol{x}') + \hat{L} \cdot r$.

6. Computational Experiments

We conduct experiments on publicly available ReLU networks to validate the performance of our method in retrieving realistic solutions with high objective values. Experiments were performed using Gurobi 11.0 (Gurobi Optimization, LLC (2023)) and PyTorch 2.0 (Paszke et al. (2019)) over Python 3.9.18 (Van Rossum & Drake (2009)) and were executed on an internal cluster with a 2.20GHz Intel(R) Xeon(R) Gold 5120 CPU and 32 GB memory.

6.1. Pretrained Neural Networks

We use publicly available feed-forward ReLU networks pretrained on MNIST (LeCun & Cortes (2010)) and CIFAR-10 (Krizhevsky (2009)), sourced from ERAN (2020). For MNIST, each network is a function $f : [0,1]^{28\times28} \to \mathbb{R}^{10}$, where each of the 10 outputs corresponds to the logits of each digit from 0 to 9, while for CIFAR-10, each network is a function $f : [0,1]^{3\times32\times32} \to \mathbb{R}^{10}$. In the experiments, we use the $3\times50, 3\times100, 4\times1024, 5\times100, 6\times100, 6\times200, 9\times100$ and the 9×200 networks pre-trained on MNIST and the $4\times100, 6\times100, 7\times1024$ and the 9×200 networks pre-trained on CIFAR-10. In the network name, the first number denotes the number of hidden layers, and the second the number of neurons per hidden layer. The accuracy in the test set and the number of total parameters per network are provided in Figure 7 in section G of the Appendix.

6.2. Experimental Setup

For each pre-trained ReLU neural network, we solve the optimization problems $\max_{\boldsymbol{x}\in\mathcal{P}} f(\boldsymbol{x})_j \; s.t. \; \boldsymbol{x} \; is \; realistic$, where f denotes the pre-trained ReLU network, $f(\boldsymbol{x})_j, j \in \{0,\ldots,9\}$ represents the j-th output of the network for input $\boldsymbol{x}$, and $\mathcal{P}$ is the domain over which we optimize ($\mathcal{P} = [0,1]^{784}$ for MNIST, $\mathcal{P} = [0,1]^{3,072}$ for CIFAR-10). In total, we solve $10 \cdot (8+4) = 120$ optimization problems.

6.3. Benchmarks

We compare our method against state-of-the-art MIP-based approaches and stochastic optimization algorithms.

Specifically, we consider the original MIP formulation proposed by Fischetti & Jo (2018), presented in equation (3), the Big-M+cuts formulation introduced by Anderson et al.

Table 1. Number of times per network each method finds the realistic solution with the highest objective. We report in bold the highest number per network. NA means that the method did not find a solution during the time window. Top table is MNIST, bottom is CIFAR-10.

	3×50	3×100	4×1024	5×100	6×100	6×200	9×100	9×200	Total
MIP	0	0	NA	NA	NA	NA	NA	NA	0
PGA	**5**	3	2	2	3	2	1	2	20
Simulated Annealing	0	0	0	0	0	0	0	0	0
SPSA	0	0	0	0	0	0	0	0	0
Genetic	0	0	0	0	0	0	0	0	0
AdSampling (ours)	**5**	**7**	**8**	**8**	**7**	**8**	**9**	**8**	**60**

	4×100	6×100	7×1024	9×200	Total
MIP	NA	NA	NA	NA	0
PGA	1	2	3	3	9
Simulated Annealing	0	0	0	0	0
SPSA	0	0	0	0	0
Genetic	0	0	0	0	0
AdSampling (ours)	**9**	**8**	**7**	**7**	**31**

(2020), and the $N = \{2, 4\}$ equal-size partition method by Tsay et al. (2021). We also use the following stochastic optimization algorithms. Projected Gradient Ascent with momentum (PGA; Kingma & Ba (2014)), Simulated Annealing (Kirkpatrick et al., 1983), Simultaneous Perturbation Stochastic Approximation (SPSA; Spall (1992)) and Genetic Algorithm (Gad, 2021; Katoch et al., 2021).

To incorporate the realistic constraint, we make the following adjustments. For MIP-based methods, we use the sampling algorithm of Section 4.3, and for each sample $\boldsymbol{x}_i$ we solve an MIP with the constraint $\delta(\boldsymbol{x}, \boldsymbol{x}_i) \leq t \cdot \max\{\delta(\boldsymbol{x}_i, N_1(\boldsymbol{x}_i), d_1(N_1(\boldsymbol{x}_i))\}$. In the end, we report the realistic solution with the highest objective. Similarly, for the stochastic algorithms, at each iteration, we perform a projection to the realistic space, by applying the constraint $\delta(\boldsymbol{x}, \boldsymbol{x}_i) \leq t \cdot \max\{\delta(\boldsymbol{x}_i, N_1(\boldsymbol{x}_i), d_1(N_1(\boldsymbol{x}_i))\}$. To ensure a fair comparison across all methods, we apply a timeout of 60 seconds, a threshold $t = 1.2$ and we use the samples in the same order as used in our algorithm across all methods and experiments. In Section H of the Appendix, we provide the exact hyperparameters used for each algorithm.

6.4. Metrics

Our evaluation is twofold. First, we report the number of times per network each method finds the solution with the highest objective. Second, to validate whether the resulting solutions are indeed realistic, we use the Local Outlier Factor classifier by scikit-learn (Buitinck et al., 2013) using the default parameters. In Table 1 we report the results.

6.5. Results

We observe that our method finds the realistic solution with the highest objective in the majority of experiments. For MNIST trained networks, our method finds the solution with the highest objective in 60 out of 80 experiments, while for the CIFAR-10 trained networks finds the realistic solution with the highest objective in 31 out of 40 experiments. The second best-performing method is the PGA. Due to the addition of realistic constraints, all solutions from all methods were classified as realistic by the LOF classifier.

In terms of scalability, we observed that among all MIP-based methods, only the original MIP method could find realistic solutions for small neural networks within the given time limit. This is the reason why in the tables we do not report results for the Big-M+cuts and the N equal-size partition methods. The average optimality gap for the MIP method for the 3×50 MNIST network is 307.78% and for the 3×100 MNIST network is 1029.98%. On the other hand, we observe that our method can scale for networks with millions of parameters, similar to the stochastic algorithm benchmarks, and still retrieve realistic solutions with the highest possible objective. By exploring the carefully selected realistic polytopes, our method *efficiently* identifies the optimal realistic prescription, since over each polytope it solves optimally a tractable optimization problem (SOCP or LP). In section F of the Appendix, we extend our experiments to modern and larger deep learning architectures including the ResNet50 v1.5 model and the Vision Transformer model (ViT; 16b) trained on the ImageNet dataset (Deng et al., 2009).

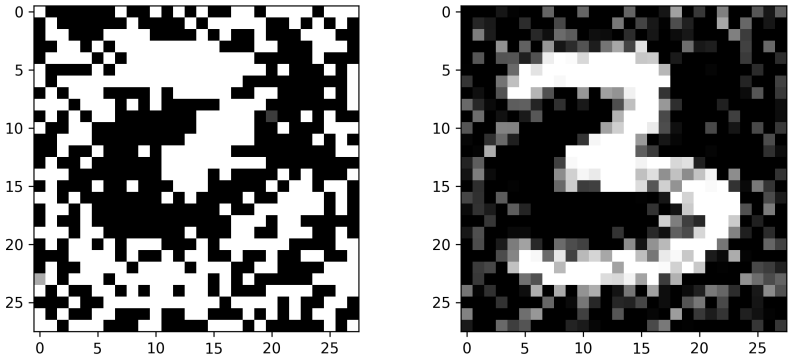

Figure 4. (Left) The optimal solution for digit 3 on network 3×50 using our method without the realistic constraint. (Right) The optimal solution for digit 3 on network 3×50 using our method with the realistic constraint.

6.6. An Example on the Effectiveness of Realistic Constraint

To demonstrate the effectiveness of the proposed realistic constraints, in Figure 4, we present the optimal solution for digit 3 by our method for the MNIST 3×50 network, with and without the realistic constraint. In Figure 4 (left) we observe that the optimal solution for digit 3 without the realistic constraint does not resemble an actual 3 and can be characterized as an outlier or an adversarial example. On the other hand, in Figure 4 (right), we observe that the optimal solution for digit 3 resembles the actual digit and thus, it can be seen that our method can indeed retrieve a realistic solution that follows the actual data manifold. We present similar visualizations on ImageNet, as well as an empirical study regarding the trade-off between realism and the user-defined threshold t in section F of the Appendix.

7. Conclusions

In this work, we focused on overcoming the Optimizer's Curse when optimizing neural networks for data-driven decision-making. We achieve this by re-formulating the highly non-linear LOF metric as a single tractable optimization constraint, thereby modeling tractably the requirement that the resulting decision should be realistic. To efficiently solve the problem for large networks with millions of parameters, we introduced an adaptive sampling algorithm. This algorithm reduces the initial hard-to-solve optimization problem into a small number of easier-to-solve problems by constraining the decision space to realistic regions and can be applied to general machine learning models. Through extensive experiments against various benchmarks over publicly available neural networks and datasets, we demonstrated the efficacy of our method in generating optimal and realistic decisions.

Acknowledgements

This work has been supported in part through a grant from the MIT-IBM Watson AI Lab.

Impact Statement

This paper presents work whose goal is to advance the field of reliable deep learning for decision-making. In this work, we propose a novel method for obtaining optimal and realistic decisions when using neural networks for data-driven decision-making. Our proposed approach is efficient, and scalable and can be extended to any differentiable machine learning model or any machine learning model that can be expressed by linear mixed-integer constraints. Since what we propose is a fundamental optimization methodology, we do not anticipate any negative societal impact by the direct application of the proposed methodology.

References

Agarap, A. F. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., and Vielma, J. P. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 183(1):3–39, 2020.

Ban, G.-Y. and Rudin, C. The big data newsvendor: Practical insights from machine learning. *Operations Research*, 67(1):90–108, 2019.

Basri, R. and Jacobs, D. Efficient representation of low-dimensional manifolds using deep networks. *arXiv preprint arXiv:1602.04723*, 2016.

Bertsimas, D., O'Hair, A., Relyea, S., and Silberholz, J. An analytics approach to designing combination chemother-

apy regimens for cancer. *Management Science*, 62(5): 1511–1531, 2016.

Bertsimas, D., Orfanoudaki, A., and Weiner, R. B. Personalized treatment for coronary artery disease patients: a machine learning approach. *Health care management science*, 23:482–506, 2020.

Boyd, S. P. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

Carbonneau, R., Laframboise, K., and Vahidov, R. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, 184(3):1140–1154, 2008.

Chen, X., Owen, Z., Pixton, C., and Simchi-Levi, D. A statistical learning approach to personalization in revenue management. *Management Science*, 68(3):1923–1937, 2022.

Cohen, M. C., Lobel, I., and Paes Leme, R. Feature-based dynamic pricing. *Management Science*, 66(11):4921–4943, 2020.

De Palma, A., Behl, H. S., Bunel, R., Torr, P., and Kumar, M. P. Scaling the convex barrier with active sets. In *Proceedings of the ICLR 2021 Conference*. Open Review, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dutta, S., Long, J., Mishra, S., Tilli, C., and Magazzeni, D. Robust counterfactual explanations for tree-based ensembles. In *International Conference on Machine Learning*, pp. 5742–5756. PMLR, 2022.

ERAN. ERAN: ETH Robustness Analyzer for Neural Networks., 2020. URL https://github.com/eth-sri/eran.

Ettl, M., Harsha, P., Papush, A., and Perakis, G. A data-driven approach to personalized bundle pricing and recommendation. *Manufacturing & Service Operations Management*, 22(3):461–480, 2020.

Fairley, M., Scheinker, D., and Brandeau, M. L. Improving the efficiency of the operating room environment with an optimization and machine learning model. *Health care management science*, 22(4):756–767, 2019.

Ferreira, K. J., Lee, B. H. A., and Simchi-Levi, D. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & service operations management*, 18(1):69–88, 2016.

Fischetti, M. and Jo, J. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018.

Gad, A. F. Pygad: An intuitive genetic algorithm python library. *CoRR*, abs/2106.06158, 2021. URL https://arxiv.org/abs/2106.06158.

Glaeser, C. K., Fisher, M., and Su, X. Optimal retail location: Empirical methodology and application to practice: Finalist–2017 m&som practice-based research competition. *Manufacturing & Service Operations Management*, 21(1):86–102, 2019.

Guidotti, R. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pp. 1–55, 2022.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com.

Huchette, J., Muñoz, G., Serra, T., and Tsay, C. When deep learning meets polyhedral theory: A survey. *arXiv preprint arXiv:2305.00241*, 2023.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Kanamori, K., Takagi, T., Kobayashi, K., and Arimura, H. Dace: Distribution-aware counterfactual explanation by mixed-integer linear optimization. In *IJCAI*, pp. 2855–2862, 2020.

Katoch, S., Chauhan, S. S., and Kumar, V. A review on genetic algorithm: past, present, and future. *Multimedia tools and applications*, 80:8091–8126, 2021.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification:*

29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30, pp. 97–117. Springer, 2017.

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljić, A., et al. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31*, pp. 443–452. Springer, 2019.

Kim, H., Papamakarios, G., and Mnih, A. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pp. 5562–5571. PMLR, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. Optimization by simulated annealing. *science*, 220(4598): 671–680, 1983.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015.

Lee, G.-H., Alvarez-Melis, D., and Jaakkola, T. S. Towards robust, locally linear deep networks. *arXiv preprint arXiv:1907.03207*, 2019.

Lin, T. and Zha, H. Riemannian manifold learning. *IEEE transactions on pattern analysis and machine intelligence*, 30(5):796–809, 2008.

Liu, H., Li, Z., Hall, D., Liang, P., and Ma, T. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *CoRR*, abs/2305.14342, 2023. doi: 10.48550/ARXIV.2305.14342. URL https://doi.org/10.48550/arXiv.2305.14342.

Liu, X., Han, X., Zhang, N., and Liu, Q. Certified monotonic neural networks. *Advances in Neural Information Processing Systems*, 33:15427–15438, 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lucic, A., Oosterhuis, H., Haned, H., and de Rijke, M. Focus: Flexible optimizable counterfactual explanations for tree ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 5313–5322, 2022.

McDonald, T., Tsay, C., Schweidtmann, A. M., and Yorke-Smith, N. Mixed-integer optimisation of graph neural networks for computer-aided molecular design. *arXiv preprint arXiv:2312.01228*, 2023.

Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heintz, I., and Roth, D. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.

Mišić, V. V. Optimization of tree ensembles. *Operations Research*, 68(5):1605–1624, 2020.

Osher, S., Shi, Z., and Zhu, W. Low dimensional manifold model for image processing. *SIAM Journal on Imaging Sciences*, 10(4):1669–1690, 2017.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library, 2019.

Perakis, G. and Tsiourvas, A. Optimizing objective functions from trained relu neural networks via sampling, 2022.

Serra, T., Tjandraatmadja, C., and Ramalingam, S. Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, pp. 4558–4566, 2018.

Serra, T., Kumar, A., and Ramalingam, S. Lossless compression of deep neural networks. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 417–430. Springer, 2020.

Smith, J. E. and Winkler, R. L. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006.

Spall, J. C. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 1992.

Strong, C. A., Wu, H., Zeljić, A., Julian, K. D., Katz, G., Barrett, C., and Kochenderfer, M. J. Global optimization of objective functions represented by relu networks. *Machine Learning*, 112(10):3685–3712, 2023.

Sun, W. and Tsiourvas, A. Learning prescriptive relu networks. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 33044–33060. PMLR, 2023.

Tjeng, V., Xiao, K., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Tong, J., Cai, J., and Serra, T. Optimization over trained neural networks: Taking a relaxing walk. *arXiv preprint arXiv:2401.03451*, 2024.

Tsay, C., Kronqvist, J., Thebelt, A., and Misener, R. Partition-based formulations for mixed-integer optimization of trained relu neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Tsiourvas, A., Sun, W., and Perakis, G. Manifold-aligned counterfactual explanations for neural networks. In Dasgupta, S., Mandt, S., and Li, Y. (eds.), *International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain*, volume 238 of *Proceedings of Machine Learning Research*, pp. 3763–3771. PMLR, 2024. URL https://proceedings.mlr.press/v238/tsiourvas24a.html.

Van Rossum, G. and Drake, F. L. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

A. Proof of Theorem 4.3

Theorem 4.3. *Let f_θ be a ReLU network, $\mathcal{P}_j$ be a polytope of f_θ that contains at least one data point with $LOF_{1,\mathcal{D}} \leq t$, $\boldsymbol{z}^j$ be the values of $\mathcal{P}_j$'s corresponding activation pattern, $f_\theta(\cdot;\boldsymbol{z}^j)$ be the given ReLU network with fixed activation $\boldsymbol{z}^j$ and $\mathcal{D}_j := \mathcal{D} \cap \mathcal{P}_j$ be the set of points of $\mathcal{D}$ that belong to $\mathcal{P}_j$. We denote as $\boldsymbol{x}'$ the data point in $\mathcal{D}_j$ such that $LOF_{1,\mathcal{D}}(\boldsymbol{x}') \leq t$ and $f_\theta(\boldsymbol{x}') \geq f_\theta(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathcal{D}_j$. Then,*

1. *(**A Tractable Formulation**) The optimization problem*

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}\in\mathcal{P}} & f_\theta(\boldsymbol{x};\boldsymbol{z}^j) \\ \text{subject to} & \delta(\boldsymbol{x},\boldsymbol{x}') \leq t \cdot \max\{\delta(\boldsymbol{x}',\boldsymbol{x}''), d_1(\boldsymbol{x}''))\}, \\ & \boldsymbol{x}'' \in N_1(\boldsymbol{x}'), \end{array} \tag{9}$$

   retrieves the realistic solution $\boldsymbol{x}^* \in \mathcal{P}_j$ with the highest objective, $f_\theta(\boldsymbol{x}^*;\boldsymbol{z}^j) \geq f_\theta(\boldsymbol{x}';\boldsymbol{z}^j)$.

2. *(**A Tight Upper Bound on the Objective**) The objective value of the optimal solution of problem (6) is upper bounded as $f_\theta(\boldsymbol{x}^*;\boldsymbol{z}^j) \leq (\boldsymbol{w}^j)^T(\boldsymbol{x}' + r\frac{\boldsymbol{w}^j}{||\boldsymbol{w}^j||_2})$, where $\boldsymbol{w}^j \in \mathbb{R}^d$ is the vector of the weights that define the hyperplane of f_θ over $\mathcal{P}_j$ and $r := t \cdot \max\{\delta(\boldsymbol{x}',\boldsymbol{x}''), d_1(\boldsymbol{x}''))\},\ \boldsymbol{x}'' \in N_1(\boldsymbol{x}')$. The bound is tight when $\boldsymbol{x}^* = \boldsymbol{x}' + r\frac{\boldsymbol{w}^j}{||\boldsymbol{w}^j||_2}$.*

Proof. 1. ***(A Tractable Formulation)*** Let a $\boldsymbol{x} \in \mathcal{P}_j$ starting at $\boldsymbol{x}' \in \mathcal{D}_j$. Due to the piecewise linear and, therefore, monotonic structure of f_θ over $\mathcal{P}_j$, we can gradually start from the inlier $\boldsymbol{x}'$ and move $\boldsymbol{x}$ towards the stationary point/maximizer $\tilde{\boldsymbol{x}}_j$ of f_θ in $\mathcal{P}_j$ through the direction defined by $\boldsymbol{x}'$ and $\tilde{\boldsymbol{x}}_j$ until reaching the furthest position such that $\boldsymbol{x}$ is still an inlier. In order for $\boldsymbol{x}$ to be an inlier, we require

$$LOF_{1,\mathcal{D}}(\boldsymbol{x}) \leq t \implies lrd_1(\boldsymbol{x}') \leq t \cdot lrd_1(\boldsymbol{x}) \implies \max\{\delta(\boldsymbol{x},\boldsymbol{x}'), d_1(\boldsymbol{x}')\} \leq t \cdot \max\{\delta(\boldsymbol{x}',\boldsymbol{x}''), d_1(\boldsymbol{x}'')\}, \tag{10}$$

where $\boldsymbol{x}'' \in N_1(\boldsymbol{x}')$. By observing that $d_1(\boldsymbol{x}') = \delta(\boldsymbol{x}',\boldsymbol{x}'')$, we obtain that

$$LOF_{1,\mathcal{D}}(\boldsymbol{x}) \leq t \implies \delta(\boldsymbol{x},\boldsymbol{x}') \leq t \cdot \max\{\delta(\boldsymbol{x}',\boldsymbol{x}''), d_1(\boldsymbol{x}'')\},\ \boldsymbol{x}'' \in N_1(\boldsymbol{x}'). \tag{11}$$

Therefore, by incorporating this constraint into the optimization subproblem with fixed activations $\boldsymbol{z}^j$, we allow $\boldsymbol{x}$ to move towards all possible directions starting from $\boldsymbol{x}'$ and still be a certified inlier. Furthermore, by observing that a feasible solution to (6) is $\boldsymbol{x} = \boldsymbol{x}'$, we obtain that for the optimal solution $\boldsymbol{x}^*$ we will have that $f_\theta(\boldsymbol{x}^*) \geq f_\theta(\boldsymbol{x}') \geq f_\theta(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathcal{D}_j$. □

2. ***(A Tight Upper Bound on the Objective)*** Starting from $\boldsymbol{x}'$, by relaxing the constraint that $\boldsymbol{x}$ should belong in $\mathcal{P}_j$, problem (6) reduces to

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}} & (\boldsymbol{w}^j)^T\boldsymbol{x} \\ \text{subject to} & ||\boldsymbol{x} - \boldsymbol{x}'||_2 \leq r, \end{array} \tag{12}$$

which is equivalent to

$$\begin{array}{ll} \max\limits_{\boldsymbol{\epsilon}} & (\boldsymbol{w}^j)^T(\boldsymbol{x}' + \boldsymbol{\epsilon}) \\ \text{subject to} & ||\boldsymbol{\epsilon}||_2 \leq r. \end{array} \tag{13}$$

We solve (13) using KKT conditions (Boyd & Vandenberghe, 2004). This is done by defining the Lagrangian function $\mathcal{L}(\boldsymbol{\epsilon},\lambda) := (\boldsymbol{w}^j)^T(\boldsymbol{x}' + \boldsymbol{\epsilon}) + \lambda(r^2 - \boldsymbol{\epsilon}^T\boldsymbol{\epsilon})$, taking the necessary conditions, i.e, $\begin{cases} \partial_{\boldsymbol{\epsilon}^*}\mathcal{L}(\boldsymbol{\epsilon}^*,\lambda) = 0 \implies \boldsymbol{w}^j - 2\lambda\boldsymbol{\epsilon}^* = 0, \\ \lambda(r^2 - (\boldsymbol{\epsilon}^*)^T\boldsymbol{\epsilon}^*) = 0, \\ \lambda \geq 0, ||\boldsymbol{\epsilon}^*||_2^2 \leq r^2, \end{cases}$ solving the system of equations and obtaining (assuming $\boldsymbol{w}^j \neq \boldsymbol{0}$) that $\boldsymbol{\epsilon}^* = \boldsymbol{w}^j/2\lambda$, $\lambda = ||\boldsymbol{w}^j||_2/2r$ and thus, $\boldsymbol{\epsilon}^* = r\frac{\boldsymbol{w}^j}{||\boldsymbol{w}^j||_2}$. As a result, the optimal objective value problem (6) is upper bounded by$(\boldsymbol{w}^j)^T(\boldsymbol{x}' + r\frac{\boldsymbol{w}^j}{||\boldsymbol{w}^j||_2})$. The equality holds when $\boldsymbol{x}^* = \boldsymbol{x}' + r\frac{\boldsymbol{w}^j}{||\boldsymbol{w}^j||_2} \in \mathcal{P}_j$ and therefore, problem (6) produces the same solution as problem (12). □ □

B. Proof of Proposition 4.6

Proposition 4.6. *The proposed sampling algorithm retrieves the optimal realistic solution using at most*

$$n \leq \min \left\{ |D|, \sum_{(j_1,\ldots,j_L)\in J} \prod_{l=1}^{L} \binom{n_l}{j_l} \right\} \tag{14}$$

samples, where $J = \{(j_1, \ldots, j_L) \in \mathbb{Z}^L : 0 \leq j_L \leq \min\{n_0, n_1 - j_1, \ldots, n_{l-1} - j_{l-1}\}, \forall l = 1, \ldots, L\}\}$.

Proof. Given that there are $|\mathcal{R}|$ realistic polytopes to search over and that the algorithm samples without replacement, the number of samples n is upper bounded by $|\mathcal{R}|$. $|\mathcal{R}|$ is upper bounded by both the size of the dataset $\mathcal{D}$, since in an extreme case each sample may correspond to a different realistic polytope, and by the number of total polytopes of the network N. The former usually occurs when the network is overparameterized and thus, the number of total polytopes is way larger than the number of data points, and the latter occurs when the network is under-parameterized.

By invoking the result of Serra et al. (2018), N is upper bounded by $\sum_{(j_1,\ldots,j_L)\in J} \prod_{l=1}^{L} \binom{n_l}{j_l}$, where $J = \{(j_1, \ldots, j_L) \in \mathbb{Z}^L : 0 \leq j_L \leq \min\{n_0, n_1 - j_1, \ldots, n_{l-1} - j_{l-1}\}, \forall l = 1, \ldots, L\}\}$. Therefore, by considering both cases, we have $n \leq \min \left\{ |D|, \sum_{(j_1,\ldots,j_L)\in J} \prod_{l=1}^{L} \binom{n_l}{j_l} \right\}$. □ □

C. Reduction of the Search Space by the Upper Bound of Theorem 4.3.2

We conduct an experiment where we count, for each ReLU network considered in the paper, the number of realistic polytopes both before and after using the upper bound outlined in Theorem 4.3.2. We obtain Table 2 presented below.

Table 2. Comparison of the number of realistic polytopes before and after applying the upper bound of Theorem 4.3.2. The number of realistic polytopes after applying the upper bound varies since for each output of the network, the vector of weights that defines the hyperplane of the network over a realistic polytope is different. In the experiments we used $t = 1$.

Model	N_r before	N_r after
MNIST 3×50	$22,029$	$941 - 2,384$
MNIST 3×100	$22,158$	$926 - 2,772$
MNIST 4×1024	$22,160$	$800 - 2,137$
MNIST 5×100	$22,160$	$242 - 1,210$
MNIST 6×100	$22,160$	$979 - 2,329$
MNIST 6×200	$22,160$	$1,089 - 2,075$
MNIST 9×100	$22,160$	$924 - 1,807$
MNIST 9×200	$22,160$	$592 - 1,911$
CIFAR-10 4×100	$4,282$	$56 - 358$
CIFAR-10 6×100	$4,282$	$49 - 156$
CIFAR-10 $7 \times 1,024$	$4,282$	$4 - 292$
CIFAR-10 9×200	$4,282$	$31 - 291$

By employing the upper bound of Theorem 4.3.2, the search space is significantly restricted, by 87.5% to 97.3% for MNIST and by 91.6% to 99.9% for CIFAR-10. We observe that for the larger dataset and networks of CIFAR-10, the number of realistic polytopes is lower, and the percentage of search space reduction due to the use of the bound from Theorem 4.3.2 is higher. This is one potential reason that explains why our method scales so well for the larger CIFAR-10 ReLU networks.

D. On the Selection of LOF as the Realistic Metric

In this paper, to model the constraint x *is realistic*, we use the LOF metric. We use LOF for the following reasons:

- First, compared to other distance metrics, LOF can be used to first filter out data points that are outliers, and thus, as shown in the paper, help reduce significantly the search space by restricting the domain to realistic polytopes.

- Second, due to its structure (LOF uses the max operator) LOF can be re-formulated as a single tractable constraint, as shown in Theorem 4.3.1, that only depends on the distance metric that is used (typically ℓ_p norm).
- Finally, due to its wide use in the literature, it is a metric that many researchers and practitioners trust and use. This can be seen by the total number of citations of the paper by (Breunig et al., 2000) which is almost 10,000.

In the manifold alignment literature, Wasserstein-p distance is often used because it measures the similarity between probability distributions. We did not use the Wasserstein-p distance as it is used to measure the distance between distributions and in this work, the output of the optimization problem is a single decision vector (and not a distribution). Moreover, there does not exist an empirical threshold for the Wasserstein distance, whereas in LOF we know that if it is less or equal to t, where t is a value close to 1, then the resulting decision is an inlier.

However, if we assume that the output $\boldsymbol{x}$ of the optimization problem is a degenerate distribution (Dirac delta on $\boldsymbol{x}$, i.e. $\delta_{\boldsymbol{x}}$), then the Wasserstein-p distance between the output and the empirical distribution of the dataset $\mathcal{D}$, let Q, is equal to

$$\mathcal{W}_p(\delta_{\boldsymbol{x}}, Q) = \big(\frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} ||\boldsymbol{x}_i - \boldsymbol{x}||^p\big)^{1/p}, \tag{15}$$

i.e. reduces to the norm between the resulting decision and the empirical distribution. This distance is similar to LOF (after the reformulation with the ℓ_p norm) with a main difference. Distance (15) takes into consideration all of the samples of $\mathcal{D}$, while LOF uses only the realistic sample of $\mathcal{D}$ per polytope that also achieves the high objective value. Thus, our proposed LOF-based distance, other than the benefits described at the beginning of this section, avoids using outliers that may exist in $\mathcal{D}$ and is also more efficient to compute.

E. Extension to other Machine Learning Models

E.1. Linear Regression

We denote the linear regression model as the function $f_\theta : \mathcal{P} \to \mathbb{R}$, where $\theta = \{\boldsymbol{w}, b\}$, with $\boldsymbol{w} \in \mathbb{R}^d$ to be the vector of weights and $b \in \mathbb{R}$ to be the intercept. In the case of linear regression, there is only one realistic polytope that is equal to the decision space $\mathcal{P}$. Therefore, in this case, if we denote as $\boldsymbol{x}'$ the realistic data point in $\mathcal{D}$ with the highest value of f_θ, the optimization problem

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}\in\mathcal{P}} & \boldsymbol{w}^T\boldsymbol{x} \\ \text{subject to} & \delta(\boldsymbol{x}, \boldsymbol{x}') \le t \cdot \max\{\delta(\boldsymbol{x}', \boldsymbol{x}''), d_1(\boldsymbol{x}''))\},\ \boldsymbol{x}'' \in N_1(\boldsymbol{x}'). \end{array} \tag{16}$$

retrieves the certified realistic solution $\boldsymbol{x}^* \in \mathcal{P}$ with the highest objective. Depending on the distance function used, problem (16) is either an LP or a SOCP.

E.2. Tree Ensembles

Following the notation and the optimization formulation introduced by Mišić (2020), optimizing over a tree ensemble that is used for regression can be expressed by the following MIP

$$\begin{array}{ll} \max\limits_{\boldsymbol{x}\in\mathcal{P},\boldsymbol{y}} & \sum\limits_{t=1}^{T}\sum\limits_{\ell\in\text{leaves}(t)} \lambda_t \cdot p_{t,\ell} \cdot y_{t,\ell} \\ \text{subject to} & \sum\limits_{\ell\in\text{leaves}(t)} y_{t,\ell} = 1,\ \forall t \in [T], \\ & \sum\limits_{\ell\in\text{left}(s)} y_{t,\ell} \le \sum\limits_{j\in C(s)} x_{V(s),j},\ \forall t \in [T], s \in \text{splits}(t), \\ & \sum\limits_{\ell\in\text{right}(s)} y_{t,\ell} \le 1 - \sum\limits_{j\in C(s)} x_{V(s),j},\ \forall t \in [T], s \in \text{splits}(t), \\ & \sum\limits_{j=1}^{K_i} x_{i,j} = 1,\ \forall i \in \mathcal{C}, \\ & x_{i,j} \le x_{i,j+1},\ \forall i \in \mathcal{N},\ j \in [K_i - 1], \\ & x_{i,j} \in \{0,1\},\ \forall i \in [n],\ j \in [K_i], \\ & y_{t,\ell} \in \{0,1\},\ \forall t \in [T],\ \ell \in \text{leaves}(t), \end{array} \tag{17}$$

where T is the number of decision trees in the ensemble, leaves(t) is the set of the indices of the leaves of tree t, λ_t is the weight given in the prediction of tree t, $p_{t,\ell}$ is the prediction that tree t makes when an observation reaches leaf ℓ, $y_{t,l}$ is the binary decision variable that is 1 if the observation encoded by $\boldsymbol{x}$ falls into leaf ℓ of tree t and 0 otherwise, splits(s) is the set of splits of tree t (non-terminal nodes), left(s) is the set of leaves that are accessible from the left branch, right(s) is the set of leaves that are accessible from the right branch, $C(s)$ is the set of values of variable i that participate in the split query of s, $V(s)$ is the variable that participates in split s, $\mathcal{C}$ is the set of indices of categorical variables, K_i is the discrete number of values the categorical variable i can take and $\mathcal{N}$ is the set of numerical variables.

In formulation (17), the binary variables that define each feasible polytope are the $\boldsymbol{y}$ variables. If we assume that $\mathcal{P}_k$ is a polytope of the tree ensemble that contains at least one data point with $LOF_{1,\mathcal{D}} \leq t$, $\boldsymbol{y}^k$ are the values of $\mathcal{P}_k$'s corresponding activation pattern, $\mathcal{D}_k := \mathcal{D} \cap \mathcal{P}_k$ is the set of points of $\mathcal{D}$ that belong to $\mathcal{P}_k$ and $\boldsymbol{x}'$ the data point in $\mathcal{D}_k$ such that $LOF_{1,\mathcal{D}}(\boldsymbol{x}') \leq t$ with the highest predicted value by the tree ensemble in $\mathcal{P}_k$ then, problem (17) becomes

$$
\begin{aligned}
\max_{\boldsymbol{x}\in\mathcal{P}} \quad & \sum_{t=1}^{T} \sum_{\ell \in \text{leaves}(t)} \lambda_t \cdot p_{t,\ell} \cdot y_{t,\ell}^k \\
\text{subject to} \quad & \sum_{\ell \in \text{left}(s)} y_{t,\ell}^k \leq \sum_{j \in C(s)} x_{V(s),j}, \ \forall t \in [T], s \in \text{splits}(t), \\
& \sum_{\ell \in \text{right}(s)} y_{t,\ell}^k \leq 1 - \sum_{j \in C(s)} x_{V(s),j}, \ \forall t \in [T], s \in \text{splits}(t), \\
& \sum_{j=1}^{K_i} x_{i,j} = 1, \ \forall i \in \mathcal{C}, \\
& \delta(\boldsymbol{x}, \boldsymbol{x}') \leq t \cdot \max\{\delta(\boldsymbol{x}', \boldsymbol{x}''), d_1(\boldsymbol{x}''))\}, \ \boldsymbol{x}'' \in N_1(\boldsymbol{x}'), \\
& x_{i,j} \leq x_{i,j+1}, \ \forall i \in \mathcal{N}, \ j \in [K_i - 1], \\
& x_{i,j} \in \{0, 1\}, \ \forall i \in [n], \ j \in [K_i].
\end{aligned}
\tag{18}
$$

Given that $\boldsymbol{y}^k$ are already fixed, problem (18) is either an LP or a SOCP, depending on the norm used as distance δ.

Remark E.1. For $T = 1$, the problem reduces to retrieving the realistic solution from a single decision tree.

F. Experiments on Modern Deep Learning Architectures

We extend our experiments to modern and larger deep learning architectures. We examine the ResNet50 v1.5 model and the Vision Transformer model (ViT; 16b) which are both trained on ImageNet 2012-2017 image classification and localization dataset (Deng et al., 2009). This dataset spans 1000 object classes and contains 1,281,167 training images, 50,000 validation images, and 100,000 test images. ResNet50 v1.5 model has a top-1 accuracy of 76.13%, a top-5 accuracy of 92.862%, and 25.5m trainable parameters. ViT has a top-1 accuracy of 85.304%, a top-5 accuracy of 97.65%, and 86.9m trainable parameters. We use the proposed methodology using the animal/dog image subset of ImageNet. Our goal is to find realistic images with the highest probability of being classified as a *Labrador Retriever*.

We perform 5 runs per model using the methodology that was described in section 5 with $t \in \{1, 1.1, 1.2\}$, a learning rate equal of 0.01, and max number of iterations equal to 10,000. First, we report the average time to convergence per model to verify the scalability of the proposed method. Second, to validate whether the resulting solutions are indeed realistic, we use the Local Outlier Factor classifier by scikit-learn (Buitinck et al., 2013) using the default parameters. We run the experiments using an NVIDIA GeForce RTX 3090 GPU. We observe that the average time for our method to convergence for ResNet50 v1.5 is just ≈ 2 minutes and for ViT is ≈ 3 mins. The proposed methodology can scale for modern deep learning architectures and produce realistic and optimal solutions in a couple of minutes, even when using commercial GPUs. As in section 6, all solutions when we used the realistic constraint were classified as realistic by the LOF classifier. In Figures 5 and 6 we present examples that demonstrate the effectiveness of our method for different values of t.

We observe that in both cases, the addition of the realistic constraint leads to solutions with high objective value that resemble actual labrador retrievers. On the other hand, removing the realistic constraint leads to solutions that have a very high objective value (high probability of being classified as labrador retriever) but do not resemble an actual labrador. Interestingly, we observe a trade-off between realism and the user-defined threshold t. The larger the value of t is, the higher the final objective value will be (since we allow the projected gradient ascent to search in a larger domain), but the final solution will be less realistic. On the other hand, the lower the value of t is, the less the final objective value will be, but the final solution will be more realistic. This is observed in Figures 5 and 6 where images with higher user-defined threshold look more distorted.

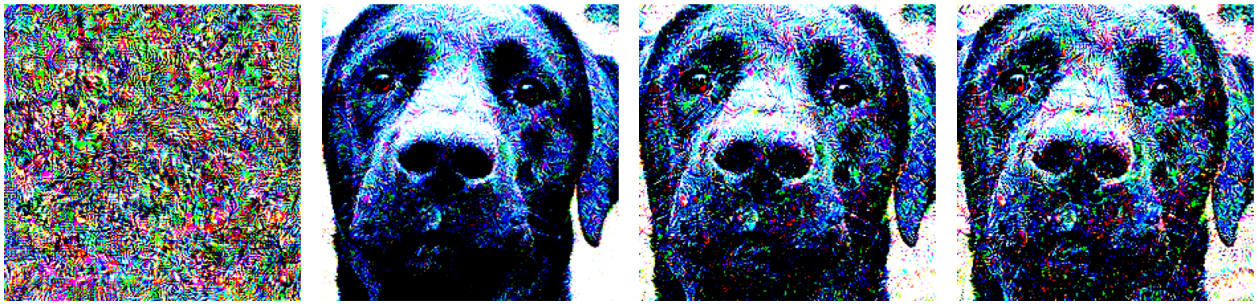

Figure 5. ResNet50: (a) Optimal not realistic solution (b) Realistic solution $t = 1$ (c) Realistic solution $t = 1.1$ (d) Realistic solution $t = 1.2$

Figure 6. ViT: (a) Optimal not realistic solution (b) Realistic solution $t = 1$ (c) Realistic solution $t = 1.1$ (d) Realistic solution $t = 1.2$

G. Accuracy and Number of Parameters of the Pre-Trained ReLU Networks

In the experiments, we use ReLU networks from ERAN pre-trained on MNIST and CIFAR-10 datasets. The accuracy of each network in the test set and the number of total trainable parameters per network are provided in Figure 7 below.

H. Hyperparameters

As mentioned in the experiments section, we report the combination of hyperparameters used for each method.

- MIP (Fischetti & Jo, 2018): `MIPGap` $= 10^{-4}$, `FeasibilityTol` $= 10^{-3}$, `OptimalityTol` $= 10^{-3}$, `BarConvTol` $= 10^{-2}$.
- Big-M+cuts (Anderson et al., 2020): `MIPGap` $= 10^{-4}$, `Cuts` $= 0$, `PreCrush` $= 1$, `FeasibilityTol` $= 10^{-3}$, `OptimalityTol` $= 10^{-3}$, `BarConvTol` $= 10^{-2}$.
- N equal-size partition (Tsay et al., 2021): `MIPGap` $= 10^{-4}$, `Cuts` $= 1$, `MIPFocus` $= 3$, `Method` $= 1$, `FeasibilityTol` $= 10^{-3}$, `OptimalityTol` $= 10^{-3}$, `BarConvTol` $= 10^{-2}$.
- PGA (Kingma & Ba, 2014): `optimizer` = Adam, `n_iterations` $= 20000$, `lr` $= 10^{-3}$, `tolerance` $= 10^{-4}$.
- Simulated Annealing (Kirkpatrick et al., 1983): `n_iterations` $= 20000$, `temperature` $= 20$, `step size` $= 2.5 \cdot 10^{-3}$.
- SPSA (Spall, 1992): `n_iterations` $= 20000$, `lr` $= 10^{-3}$.
- Genetic Algorithm (Gad, 2021; Katoch et al., 2021): `num_generations` $= 800$, `num_parents_mating` $= 32$, `sol_per_pop` $= 32$, `parent_selection_type` = sss, `keep_parents` $= 8$, `crossover_type` $=$ single point, `mutation_type` $=$ random, `mutation_percent_genes` $= 10$.
- AdSampling (ours): `MIPGap` $= 10^{-4}$, `FeasibilityTol` $= 10^{-3}$, `OptimalityTol` $= 10^{-3}$, `BarConvTol` $= 10^{-2}$.

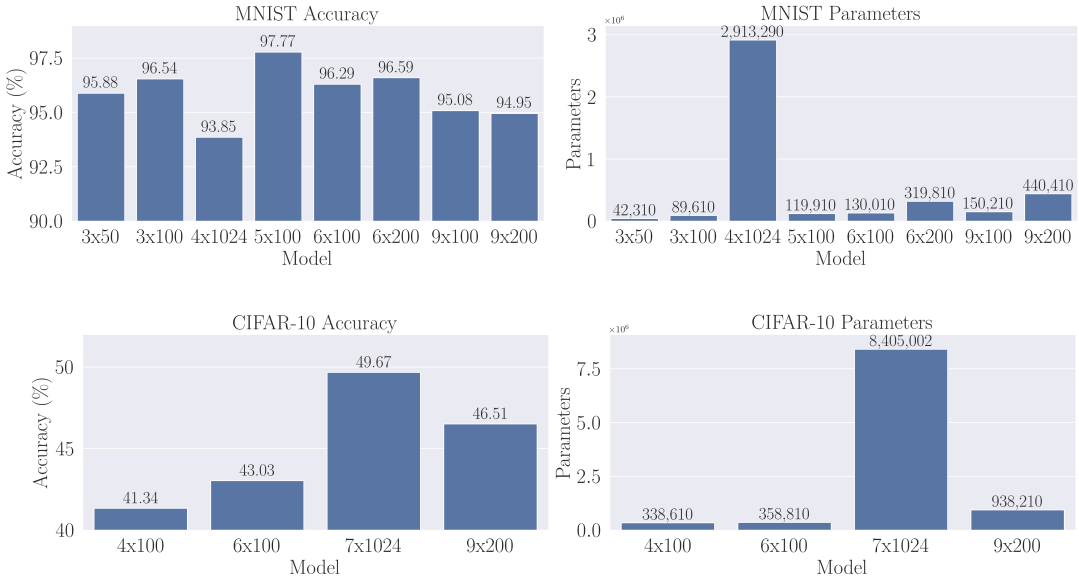

Figure 7. Test set accuracy and number of trainable parameters per network for the MNIST and CIFAR-10 datasets.

All methods were given a time window of 60 seconds.

For the MIP-based methods, instead of using big-M for the upper and lower bounds $\boldsymbol{u}_l$ and $\boldsymbol{l}_l$, to speed up the computations we used the tighter bounds proposed by Liu et al. (2020). Liu et al. (2020) refined those vectors by calculating tighter bounds using LP as a pre-processing step. Specifically, the authors suggested calculating tighter bounds as $\boldsymbol{u}_l = \sup_{\boldsymbol{l}_{l-1} \leq \boldsymbol{x}_{l-1} \leq \boldsymbol{u}_{l-1}} \{\boldsymbol{W}_l \boldsymbol{x}_{l-1} + \boldsymbol{b}_l\}$ and $\boldsymbol{l}_l = \inf_{\boldsymbol{l}_{l-1} \leq \boldsymbol{x}_{l-1} \leq \boldsymbol{u}_{l-1}} \{\boldsymbol{W}_l \boldsymbol{x}_{l-1} + \boldsymbol{b}_l\}$. For $\boldsymbol{x}_0$, the upper and lower bounds are obtained from $\mathcal{P}$.