PICE: Polyhedral Complex Informed Counterfactual Explanations

Mattia Jacopo Villani[1,2*], Emanuele Albini[2], Shubham Sharma[2], Saumitra Mishra[2], Salim Ibrahim Amoukou[2], Daniele Magazzeni[2], Manuela Veloso[2]

[1]King's College London
[2]J.P. Morgan Chase
mattia.villani@kcl.ac.uk, emanuele.albini@jpmorgan.com, shubham.x2.sharma@jpmorgan.com, saumitra.mishra@jpmorgan.com, salim.ibrahimamoukou@jpmorgan.com, daniele.magazzeni@jpmorgan.com, manuela.veloso@jpmorgan.com

Abstract

Polyhedral geometry can be used to shed light on the behaviour of piecewise linear neural networks, such as ReLU-based architectures. Counterfactual explanations are a popular class of methods for examining model behaviour by comparing a query to the closest point with a different label, subject to constraints. We present a new algorithm, Polyhedral-complex Informed Counterfactual Explanations (PICE), which leverages the decomposition of the piecewise linear neural network into a polyhedral complex to find counterfactuals that are provably minimal in the Euclidean norm and exactly on the decision boundary for any given query. Moreover, we develop variants of the algorithm that target popular counterfactual desiderata such as sparsity, robustness, speed, plausibility, and actionability. We empirically show on four publicly available real-world datasets that our method outperforms other popular techniques to find counterfactuals and adversarial attacks by distance to decision boundary and distance to query. Moreover, we successfully improve our baseline method in the dimensions of the desiderata we target, as supported by experimental evaluations.

1 Introduction

When neural networks are used to make decisions, such as the outcome of a loan application, or the allocation of a trading strategy, it is important to gain an understanding of how these models operates to ensure their safety. Computing explanations that are faithful (rather than approximate) to the neural network behaviour often requires an understanding of the inner architecture of the model, which is often complex and poorly understood. Continuous PieceWise Linear (CPWL) neural networks are amongst the most popular types of architectures, which include networks using as activation function Rectified Linear Unit (ReLU), the default activation function for `tensorflow` and `sklearn` feedforward architectures, but also Leaky ReLU and Parametrised ReLU, hard hyperbolic tangent, and absolute value functions (Sharma, Sharma, and Athaiya 2017; Szandała 2021). Model explainers can leverage theoretical insights to ensure that the explanations are *exact*, meaning that no approximation is involved in their computation. Such explanations are defined unequivocally, making them suitable candidates for regulatory standards.

Recently, the employment of tools from polyhedral geometry has unlocked fresh insight into the behaviour of networks with CPWL activation functions (Serra, Tjandraatmadja, and Ramalingam 2018; Hanin and Rolnick 2019; Rolnick and Kording 2020; Berzins 2023). This has enabled the emergence of methods that are used to extract the *polyhedral complex*: the collection of linear models and polyhedrons over which each model is defined, which account for an exact representation of the overall network as a CPWL function. Such advances promise to enhance interpretability and generate new model diagnostic tools to comprehend and evaluate networks in deployment, as shown by Sotoudeh and Thakur (2019) and Sudjianto et al. (2020). Specifically, reducing a piecewise linear neural networks into a collection of polyhedra with associated linear models provides a complete and exact interpretation of the model behaviour. Given a data point we are able to recover the input-output relationship in the form of a linear model. This has implications for all stakeholders: model builders can design networks that are robust to prescribed specifications, model explainers are able to derive faithful and consistent explanations, and model users may assess alignment to their desired outcomes. Moreover, focusing on faithful explanations addresses the reproducibility crisis in Artificial Intelligence (Hutson 2018) since such methods do not employ any randomness.

Counterfactual explanations (also referred to as counterfactuals) have been popularly used to explain the predictions of models (Guidotti 2022) due to their intuitive and user-friendly nature (Miller 2019). For a given classifier and a query instance, a counterfactual explanation is given by a point such that (a) the decision made by the classifier is changed and (b) the distance between the two instances is *minimal*. While the notion of minimality is nuanced and context-dependent, the original formulation by Wachter, Mittelstadt, and Russell (2017) framed the counterfactual search as an optimization problem aimed at finding the closest instance to a query in the Euclidean input space which has a different outcome from the classifier. Subsequently, several desiderata were proposed, including sparsity (Sharma, Henderson, and Ghosh 2019; Brughmans, Leyman, and Martens 2023; Virgolin and Fracaros 2023),

*Work done during an internship at J.P. Morgan Chase.

plausibility (Poyiadzi et al. 2020; Schleich et al. 2021), diversity (Mothilal, Sharma, and Tan 2020), actionability (Ustun, Spangher, and Liu 2019; Karimi, Schölkopf, and Valera 2021), and robustness (Dutta et al. 2022).

Minimal counterfactuals are highly desirable because the cost of recourse for users is often defined as function of counterfactual distance. For example, in the setting of mortgage applications, a customer may request a counterfactual explanation to improve their profile and reapply at a future moment. The counterfactual explanation may suggest that a successfully candidature would need increases in financial features, such as salary, savings, credit score. In this setting, providing minimal counterfactuals promotes financial inclusion and decreases the burden of reapplication for consumers.

However, finding provably minimal counterfactuals for neural networks is highly nontrivial. In this paper, we propose a collection of techniques to find counterfactual explanations in the context of CPWL networks. We show that in the setting of polyhedral geometry, it is possible to gain increased control on the counterfactual desiderata. In particular, this is achieved via a novel and efficient algorithm to find points that are provably the closest counterfactuals for a given query, subject to any sparsity and collection of actionability constraints. Specifically, our contributions are:

- A novel algorithm to compute counterfactuals that lie exactly on the decision boundary for CPWL networks, which we call PICE (Polyhedral-complex Informed Counterfactual Explanations), which is provably the closest possible counterfactual to a query in the Euclidean distance;
- A scalable variant of PICE (Fast) that targets **speed** and for which we prove lower bounds on the probability of convergence to the minimum counterfactual;
- Four additional variants of PICE that target the most popular counterfactual desiderata: **sparsity**, **plausibility**, **actionability**, and **robustness**, showcasing their effectiveness in real-world datasets.

In our experimental evaluation, we achieve state-of-the-art results in minimality and validity rates of our counterfactuals across datasets. Moreover, the model user can achieve great control over the counterfactual explanation desiderata that they want to pursue. We show this by comparing our desiderata-focused algorithms to counterfactual techniques that target the same desiderata, achieving better L_0 in comparison to other sparsity-pursuing techniques and improving the plausibility of our counterfactuals.

2 Related Works

Counterfactuals Techniques. In algorithmic recourse, presenting a model user with a counterfactual empowers them to understand what needs to be changed in order to achieve a desired outcome (Verma et al. 2020; Guidotti 2022). This is an example of a contrastive explanation. Several works address how to compute counterfactuals (Wachter, Mittelstadt, and Russell 2017; Pawelczyk, Broelemann, and Kasneci 2020; Van Looveren and Klaise 2021). In particular, works targeting the construction of counterfactuals include Tran, Ghazimatin, and Saha Roy (2021), in the context of recommender systems, and Yang et al. (2022), using reinforcement learning in a model agnostic setting. The closest works to our paper also leverage the piecewise linear topology of networks with ReLU networks in order to obtain counterfactuals that are closer to the decision boundary with Mixed Integer Linear Programming (Mohammadi et al. 2021), but do not guarantee minimality. Carreira-Perpiñán and Hada (2023) propose an algorithm to find counterfactuals that have provable minimum distance from the query in the context of random forests, but not for CPWL networks.

Piecewise Linear Geometry of ReLU Networks. The development of theoretical tools to understand the properties of ReLU networks is an active area of research (Eldan and Shamir 2016; Hanin and Sellke 2017; Rolnick and Kording 2020; Grigsby, Lindsey, and Rolnick 2023). For instance, efforts have been place in enumerating the number of linear regions (Montufar et al. 2014; Arora, Cohen, and Hazan 2018; Serra, Tjandraatmadja, and Ramalingam 2018; Hanin and Rolnick 2019).

Recent algorithms extract the full polyhedral complex, either by techniques known as *subdvision* (Raghu et al. 2017; Humayun, Balestriero, and Baraniuk 2022; Wang 2022; Berzins 2023), *marching* (Lei and Jia 2020) or *pattern testing* (Balestriero and LeCun 2023; Villani and Schoots 2023). In this work, we leverage Villani and Schoots (2023) and improve on their efficiency; we choose this method since it can be adapted to decompose bounded regions of the network, which we use to target counterfactual desiderata. Our work applies the key insights of this literature to the computation of counterfactuals with minimal distance and common desiderata, providing a bridge from theory into practice.

3 Background

In this section, we present the theoretical background on polyhedral geometry for ReLU neural networks. We introduce notation and a sketch of the polyhedral extraction method.

In particular, we present the correspondence between activation patterns in a neural network, encoding the data of which neurons are active in a neural network, and the polyhedra in a complex. Polyhedra are high-dimensional convex shapes in the input space that generalise the notion of a polygon. Just like polygons are defined by their faces (line segments) and can be viewed as the intersection of half-planes, polyhedra can be represented as the intersection of half-spaces generated by hyperplanes (high dimensional generalisations of planes).

The key idea is to break down the neural network into interpretable parts: every piecewise linear network can be seen as splitting up the input space into regions called *polyhedra* and applying a unique linear model to each of them. Indeed, there is a unique activation pattern for every polyhedron; however, not all activation patterns are attained by some point in the input space. Therefore, some activation

patterns are not feasible. A formal presentation follows.

Without loss of generality, we consider throughout the paper the example of ReLU networks. An L-layer neural network with neuron vector $\mathbf{n} = [n_0, n_1, n_2, ..., n_L]$ is a parameterised function $\mathcal{N} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{L+1}}$, described by the recursion:

$$\chi^{(l)} = \sigma\left(W^{(l)}\chi^{(l-1)} + b^{(l)}\right), l \in \{1, 2, ..., L\}$$

with $\chi^{(0)} = x \in \mathbb{R}^{n_0}$, $W^{(l)} \in \mathbb{R}^{n_{l-1} \times n_l}, b^{(l)} \in \mathbb{R}^{n_l}$, the ReLU activation function $\sigma(x) = \max(0, x)$ is applied element-wise, and $\mathcal{N}(x) = W^{(L+1)}\chi^{(L)} + b^{(L+1)}$ is obtained through the output layer, parametrised by $W^{(L+1)}, b^{(L+1)}$. Such a network can be thought of as a piecewise linear function; therefore, it can be represented as a collection of polyhedra (Montufar et al. 2014) $\omega \in \Omega$, each of which is given by the intersection of a collection of half-spaces,

$$\omega = \bigcap_{i \in \mathcal{I}} \{x : A_i x \leq c_i\}$$

where $\mathcal{I}$ is an indexing set and $A_i \in \mathbb{R}^{n_0}, c_i \in \mathbb{R}$ specify the hyperplane in the inequality. A bounded polyhedron is called a polytope. At each one of these polyhedra ω, we can compute exactly the coefficients of the linear model $\alpha_\omega, \beta_\omega$, describing the exact functional relationship between inputs and outputs for all points in the polyhedron Sudjianto et al. (2020).

Finding these coefficients and polyhedra for all $x \in \mathbb{R}^{n_0}$ results in an exact representation of the neural network as a lookup table: given a query, we find the polyhedron in which it resides and apply the linear model relating to that polyhedron. In other words, $\mathcal{N}$ is fully determined by a decomposition $\mathcal{C}(\mathcal{N}) = (\Omega, \{\alpha_\omega, \beta_\omega\}_{\omega \in \Omega})$, where the following holds:

$$\mathcal{N}(x) = \alpha_\omega \cdot x + \beta_\omega, \forall x \in \omega.$$

The algebraic geometrical properties of this structure are studied in (Grigsby and Lindsey 2022), where $\mathcal{C}(\mathcal{N})$ is referred to as the *polyhedral complex*.

To extract such a polyhedral complex, **pattern testing** based extraction algorithms can be used. Let $P^{(l)} \in \{0, 1\}^{n_l}$ be an *activation vector* for layer l whenever it represents the indicator of whether the neurons of a layer $\chi^{(l)}$ are greater than zero (1) or are zero (0): $P_i^{(l)} = \mathbb{I}_{\chi_i^{(l)} > 0}(x)$.

The activation vectors are living in a space $\{0, 1\}^{n_l}$ and can be thought of independently of x. Notice that every input has a unique activation vector at each layer. The collection of activation vectors for all layers is the *activation pattern* $P = [P^{(1)}, P^{(2)}, ..., P^{(L)}]$. The key idea of pattern testing algorithm is that given point $x \in \mathbb{R}^{n_0}$, the collection of points with the same activation pattern can be represented as a polyhedron ω. Because these points have the same activation pattern, the weights $\alpha_\omega, \beta_\omega$ can be explicitly found. In this sense, the activation pattern acts as a unique label for each polyhedron. Therefore, finding all polyhedra is equivalent to finding all feasible patterns. A pattern is *feasible* if the intersection of the constraints determined by its activation vectors is non-empty.

The following definition from Villani and Schoots (2023) describes the conditions for the existence of some $x \in \mathbb{R}^{n_0}$ such that a given pattern P is *feasible* in the space. Here, and everywhere in the paper, inequalities between vectors are interpreted as being satisfied element-wise for all constraints.

Definition 3.1. (Villani and Schoots 2023) An activation pattern $\{P^{(1)}, ..., P^{(L)}\}$ is feasible if there exists an $x \in \mathbb{R}^{n_0}$ such that for all $l \in [L]$,

$$\text{diag}(\mathbf{1} - 2P^{(l)}) \cdot A^{(l)} \cdot \chi^{(l-1)} \leq -\text{diag}(\mathbf{1} - 2P^{(l)}) \cdot d^{(l)}$$

where $A^{(l)}, d^{(l)}$ are linearised preactivation such that

$$\chi^{(l)} = \text{ReLU}(A^{(l)}x + d^{(l)}),$$

and they can be found recursively by:

$$\begin{aligned} A^{(1)} &:= W^{(1)} \\ A^{(l)} &:= W^{(l)} \cdot \text{diag}(P^{(l-1)}) \cdot A^{(l-1)} \\ d^{(1)} &:= b^{(1)} \\ d^{(l)} &:= W^{(l)} \cdot \text{diag}(P^{(l-1)}) \cdot d^{(l-1)} + b^{(l)}. \end{aligned}$$

This definition makes the correspondence between activation patterns and linear programs explicit, while providing a geometric interpretation of the activation pattern in the input space. For every pattern, there is a linear program that needs to be satisfied in order for the pattern to be feasible. The set of non-redundant constraints of this linear program is exactly the set of hyperplanes in which the faces of the polyhedron live. The significance of being able to represent these conditions as linear program is that we are able to leverage existing solvers to quickly verify the existence of points that satisfy the constraints, determining the feasibility of a linear pattern. We use this in our decomposition algorithm and throughout our Methodology.

4 Methodology

Our proposed method finds minimal counterfactuals for a given query.

The first step is to decompose the network, which outputs a collection of polyhedra (the polyhedral complex). This reduces the neural network to a lookup table: for each query in the input space, there is a unique polyhedron in which it falls; for each polyhedron, there is a linear model that is applied to it. This describes the neural network's function exactly.

The second step is to refine the partition by finding the decision boundary whenever it cuts through a polyhedron. This refines the lookup table, increasing the number of polyhedra (by cutting some of them) and changes the entries of the lookup table: instead of linear models, we now have labels. This process allows us to divide partition in two (or more in the multiclass classification setting) collections of polyhedra: those that are positively labeled (such that $\forall x \in \omega, \mathcal{N}(x) = 1$) and those that are negatively labeled.

In the last step, we select polyhedra from the refined partition that are positively labeled, and run the Nearest Polytope algorithm on them (Wu, Sadraddini, and Tedrake 2020). A high-level description of PICE is outlined in Algorithm 1.

Algorithm 1: PICE

Inputs $x \in \mathbb{R}^n, \mathcal{N} : \mathbb{R}^n \to \mathbb{R}^m$,
Outputs x^C
$(\Omega, \{\alpha_\omega, \beta_\omega\}_{\omega\in\Omega}) \leftarrow GlobalDecomposition(\mathcal{N})$ (Section 4: Decomposing the Network, Appendix A)
$(\Omega^\star, \{\alpha_{\omega'}, \beta_{\omega'}\}_{\omega'\in\Omega^\star}) \leftarrow DecisionBoundaryRefinement(\Omega, \{\alpha_\omega, \beta_\omega\}_{\omega\in\Omega})$ (Section 4: Refining the Partition)
$\Omega^+ \leftarrow ChoosePositiveRegions(\Omega^\star)$
$\omega^\star \leftarrow NearestPolytope(x, \Omega^+)$
$x^C \leftarrow argmin_{y\in\omega^\star} d(x, y)$

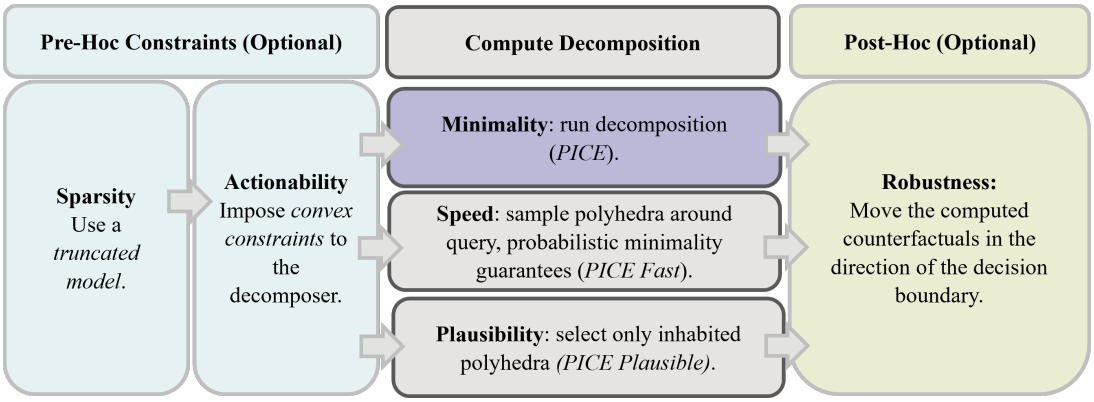

Figure 1: The PICE algorithm can be adapted to target specific counterfactual desiderata. The main algorithm (in purple) targets minimality, by computing all polyhedra in the partition and finding the closest point in L_2 space to the query. The *Fast* and *Plausible* variants decompose only the neighbouring polyhedra to the query and the inhabited polyhedra (with at least one data point for a given input dataset). To each of these variants, we can add pre-processing steps (in light blue, to the left) to pursue *Sparsity* or *Actionability*. After having found the counterfactual point, we can enforce *Robustness* (in green, to the right) post-hoc.

PICE: Decomposing the Network

To frame the problem of pattern testing as a binary tree search, we prove a new heuristic which allows us to efficiently compute the decomposition. Proving the heuristic results in a reduction of running time compared to (Villani and Schoots 2023), without compromises in the algorithm's ability to decompose the network.

Proposition 4.1 (Infeasible Subpatterns)**.** *Let $(P^{(l)}, W^{(l)}, b^{(l)})$ define a set of constraints as in Definition 3.1. Let $\mathcal{I} \subset [n_l]$ represent a subset of the indices of the constraints, such that $(P^{(l)}_\mathcal{I}, W^{(l)}_\mathcal{I}, b^{(l)}_\mathcal{I})$ represents the subset of constraints obtained by picking rows*

$$\left(diag(\mathbf{1} - 2P^{(l)}) \cdot W^{(l)} \cdot \chi^{(l-1)}\right)_\iota \leq \left(-diag(\mathbf{1} - 2P^{(l)}) \cdot b^{(l)}\right)_\iota,$$

for $\iota \in \mathcal{I}$. When these constraints are respected by $\chi^{(l-1)}$, we say that the subpattern is feasible. If a subpattern $P_\mathcal{I}$ is not feasible for $\mathcal{I}$, then any pattern containing the subpattern is not feasible.

Proposition 4.1 follows from the observation that if a set of constraints is infeasible, then adding further constraints cannot possibly yield a feasible set. At a high level, we can start from the first layer and think of each neuron as imposing a set of constraints into the input space. These constraints are specified by the weights of the linear model $W^{(1)}xb^{(1)}$. For a neuron to be active, we require that $(W^{(1)}x+b^{(1)}) > 0$ for neurons $i \in \{1, ..., n_1\}$ in the first layer, otherwise the neuron will be inactive. We want to check whether an activation vector, i.e. a specification of which neurons are active and which are inactive, is feasible.

This is the case exactly when the constraints are satisfied by some non-empty set of points. This is in exact correspondence with a linear program in convex optimisation and, consequently, a clear geometrical interpretation: the existence of a feasible activation vector is in exact correspondence with the existence of a non-empty polyhedron such that all points in that polyhedron produce the same activation vector. Checking all combinations of linear programs would require exploring 2^{n_1} instances; however, (Montufar et al. 2014) provide theoretical upper bounds that are generally lower, and (Hanin and Rolnick 2019) empirically show that the number is considerably lower.

Nevertheless, to search this space of feasible activation vectors at the first layer efficiently, we employ a binary tree search strategy. In particular, we view every branch of the tree as the state of a neuron, so that any time we find an infeasible subvector, we prune the tree and avoid testing more vector that contain the infeasible subvector. For example, if we have determined that it is impossible for the first three neurons to take the pattern $[0, 1, 1]$, we will not need to check whether it is possible for the first four neurons to have the

pattern $[0,1,1,1]$ nor $[0,1,1,0]$, and so on. Geometrically, we may think of this as the intersection of three half-spaces, which may be empty if two of the hyperplanes are parallel, for example. It is therefore redundant to check if adding a third constraint to an empty set is feasible, since the intersection of any set with the empty set must be empty. Checking at every node whether a subpattern is feasible allows us to prune the search tree, making the search itself quicker.

This procedure is applied at every layer simultanously, where feasibility of an activation vector at layer l is interpreted at the latent space in layer $l-1$, i.e. a Eucledian space $\mathbb{R}^{n_{l-1}}$. Once a set of feasible activation vector is found for each layer, we check that the activation vectors are compatible between layers. This is done by checking a global linear pattern as defined in Villani and Schoots (2023) and can be thought of geometrically as projecting the polyhedra represented by a pattern $P^{(l)}$ in layer l back into the input space to see if they intersect with each polyhedron in $P(1)$. If this is the case, then the patterns are feasible. We refer the reader to the Appendix A for a detailed exposition of the algorithms.

PICE: Refining the Partition with the Decision Boundary

Given the decomposition $\mathcal{C}(\mathcal{N})$, it is possible to find the exact points on the decision boundary:

$$D(\mathcal{N}) = \{x \in \mathbb{R}^{n_0} : |\text{argmax}(\mathcal{N}(x))| > 1\}.$$

We can then refine the partition Ω into a finer partition $\Omega^\star$ of polyhedra such that each polyhedron has a unique label.

At a high level, we can think of this procedure as assigning labels to each polyhedron, or cutting the polyhedron in parts and labeling each part with the value that the function attains at at that point. It is possible to do this because for each polyhedron we have exactly one linear model describing the input-output relationship of the function. In other words, refinement takes the representation of the network as a partition of the input with linear models defined on each part and returns a representation of the network as a finer partition, with a single label in each polyhedron, rather than a linear model.

For general architectures, finding exact solutions is an intractable problem with no simple explicit representations. However, in the piecewise linear case, we can leverage the decomposition of the network into linear maps to compute the solution exactly. This is achieved by applying a new set of constraints that replace the argmax function in the last step of computation of the network. Since these constraints are also linear, they can be represented geometrically in the input space as hyperplanes. Therefore, we can think of the refinement process as an additional feasibility check: for each polyhedron, we check which of the new constraints intersects with the polyhedron in a non-empty set.

Formally, we consider two cases: classification and regression. In the classification case, we check that a certain class $k \in \{1, ..., n_{L+1}\}$ is selected. Note that $k = \text{argmax}\mathcal{N}(x)$ is the best class exactly when $\mathcal{N}(x)_k > \mathcal{N}(x)_i, \forall i \in \{1, .., n_{L+1}\}$. In particular, we observe that this can be written as an inequality:

$$\begin{pmatrix} -1 & \dots & 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & -1 & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & -1 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 & \dots & -1 \end{pmatrix} \cdot \mathcal{N}(x) > \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

This is matrix constructed by taking $-\text{diag}(\mathbf{1}_m)$, adding a matrix with 1s on the kth column only, and removing the kth row. Observe that this set of inequalities is describing a region in the input space where the neural network will output k. Let M_k be the matrix described above, such that $M_k \cdot \mathcal{N}(x) > \mathbf{0}$; there will be exactly one such matrix for every label k. A polyhedron ω may satisfy the inequality either (1) for exactly one value of k, or (2) for multiple values of k. The first setting represents the fact that the network assigns the label k to all x in ω. The second setting implies that there are different values attained within ω. We refine the partition and label the parts accordingly, for instance $\omega_1, \omega_2, ..., \omega_m$ in the case where all values $\text{argmax}\mathcal{N}(x) = 1, ..., m$ are attained in ω.

A similar method can be applied in the regression case: by specifying a threshold τ, we check whether the output of the network (which is now real-valued, i.e. the output dimension n_{L+1}) is 1) is above this threshold $\mathcal{N}(x) > \tau$. This amounts to verifying that the half-space $\alpha_\omega x + \beta_\omega \leq \tau$ intersects the polyhedra ω for any $\omega \in \Omega$. Whenever it does, we assign a negative label on the region; we assign a positive label otehrwise.

The output of both cases is a refined partition with labels. We use this refined partition to find the minimal counterfactual.

PICE: Minimality Guarantees and Targeting Desiderata

In this section, we present our main theoretical contribution: the theorem of convergence and minimality for PICE (Algorithm 1). This provides theoretical guarantees that our method always finds an unequivocally defined exact solution to our counterfactual query. We also present variants of PICE (refer to Figure 1 for implementation details and Figure 2 for a depiction of the variants), which can be deployed to target specific desiderata.

Minimality The problem of finding the *minimal* counterfactual, which is equivalent to finding the closest point on the decision boundary to a given query x, is defined by:

$$x^C = \underset{z \in \mathbb{R}^{n_0}, \mathcal{N}(x) \neq \mathcal{N}(z)}{\text{argmin}} ||x - z||_2 = \underset{z \in D(N)}{\text{argmin}} ||x - z||_2. \quad (1)$$

PICE finds this point x^C exactly, wherever it may be in the input space. The proof of the following theorem can be found in the Appendix A.

Theorem 4.2 (PICE Convergence)**.** *For a given neural classifier $\mathcal{N} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{L+1}}$ and any query $x \in \mathbb{R}^{n_0}$ if there exists a $z \in \mathbb{R}^{n_0}$ with argmax$(\mathcal{N}(x)) \neq$ argmax$(\mathcal{N}(z))$, then Algorithm 1 always converges to the global optimal solution.*

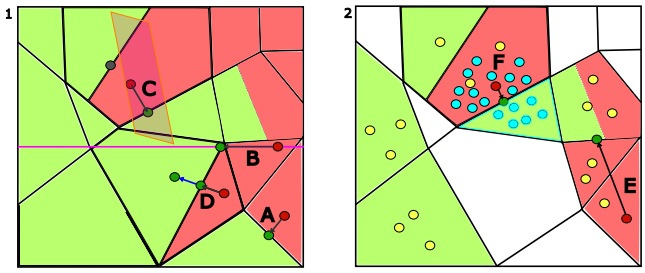

Figure 2: *PICE variants on the polyhedral complex.* Everywhere in this figure, dark red dots are queries and dark green dots are counterfactuals. Light green regions are positively labeled regions, light red regions are negatively labeled, and white regions are polyhedra that are not being considered by PICE. Dotted lines are boundaries within polyhedra that represent internal decision boundaries; arrows are counterfactual vectors, from query to counterfactual. Other lines are boundaries between polyhedra. The two dimensions represent two features, such that $\mathcal{N} : \mathbb{R}^2 \to \mathbb{R}^{n_{L+1}}$.
On the left subfigure (1) represents a refined full decomposition of the input space Ω in a bounded hypercube. Counterfactual **(A - Minimal)** targets minimality, **(B - Sparse)** is pursuing sparsity and is only allowed to vary on the subspace represented by the magenta line, but not vertically, since a direction of counterfactual is needed as input **(C - Actionable)** defines a convex actionable space on which the method can find a counterfactual, forfeiting what might be the closest. **(D - Robust)** shows the post-hoc adjustment of moving into the positive region of the input space achieved by decision boundary counterfactuals.
On the right, in subfigure (2), only a fraction of the polyhedra have been decomposed. Therefore, PICE must find a counterfactual on the polyhedra that have been found either **(E - Plausible)** by checking the inhabited polytopes (yellow points), thereby targeting plausibility, or **(F - Fast)** by sampling points around the query, targeting speed.

Speed: Sampling-Based Decomposition Decomposing the entire network can be slow, since the number of polyhedra in the input space can be large (see Montufar et al. (2014); Arora, Cohen, and Hazan (2018) for bounds and (Hanin and Rolnick 2019) for an empirical enumeration). Algorithm 3 describes a fast variant of PICE, which avoids decomposing the network. Given a query, one may find the activation pattern of a polyhedral region by running inference. The activation patterns and the weights are sufficient to compute the hyperplanes that define the polyhedron, as defined in Definitions 3.1. In this way, we search polyhedra that may lie on the other side of the decision boundary. Suppose $\mathcal{X} \subset \mathbb{R}^{n_0}$ is a collection of points sampled by Gaussian distribution with centre at x, our query, and a chosen spread σ^2. We find find all polyhedra in which at least one point lives: $\Omega^{\mathcal{X}} = \{\omega : \exists x \in \mathcal{X}, x \in \omega\}$. We label these polyhedra through refinement, determining their labels. Then we solve the nearest polytope problem on all said polyhedra that live on the other side of the decision boundary found by the sampler, as shown in Algorithm 3.

While Theorem 4.2 is no longer applicable, we provide in Theorem 4.3 lower bounds to the probability of having found the minimal counterfactual through this algorithm, when sampling in a sufficiently large hypercube around the query. Note that this is expressed through the probability of having found all of the polyhedra in a bounded partition through random sampling, when this bounded partition intersects all parts of the decomposition, which is a sufficient condition to having found the minimal counterfactual. The proof is in the Appendix C and experiments comparing speed to architecture size can be found in Appendix E.

Proposition 4.3. *Let $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}^m$ be a neural network with neuron vector $\mathbf{n} = [n_1, n_2, ..., n_L]^T, L \in \mathbb{N}$. Suppose the decomposition $(\Omega, \{\alpha_\omega, \beta_\omega\}_{\omega\in\Omega})$ is spanned by $|\mathcal{H}| \in \mathbb{N}$ hyperplanes. Let $X_1, ..., X_k \sim U([a, b]^n)$ be samples in $\mathbb{R}^n$, where a, b are sufficiently large real numbers to encompass all intersections points and such that*

$$\min_{\omega\in\Omega}\{vol(\omega \cap [a, b]^n)\} = \xi.$$

Let $m \geq |\Omega|$ be the maximum number of parts that Ω can attain. Then the probability that all regions of the partition have at least one sample $\mathbb{P}(E)$ is lower bounded by:

$$\mathbb{P}(E) \geq \sum_{i=0}^{m} \binom{m}{i} (-1)^{i-m} \left(1 - \frac{(m-i)(V + m\xi)}{mV}\right)^k,$$

where $V = \prod_{j=1}^{n}(b - a)$ is the volume of the hyperbox.

The key insight is that sampling allows us to control the probability of having found the minimal counterfactual: the more points we sample, the higher the likelihood of having found a point living in the polyhedron where the minimal counterfactual also lives. It is important to note that this method introduces randomness in the algorithm, which trades the certain minimality for speed. This method may be well suited for large architectures, as we show in Appendix E, or to precompute estimates to the minimal counterfactual. The *fast* counterfactual here is formally defined as:

$$x^C_{\text{fast}} = \underset{z\in\Omega^{\mathcal{X}}, \mathcal{N}(x)\neq\mathcal{N}(z)}{\text{argmin}} ||x - z||_2. \tag{2}$$

Convex Actionability: Bounded Decomposition In several use cases, it is important for recourse options to be actionable (Ustun, Spangher, and Liu 2019; Karimi, Schölkopf, and Valera 2021). Actionability ensures that users are effectively able to use the counterfactual explanation to change their outcomes. In the case of the user applying for a loan, it would be unreasonable to provide a recourse option where the salary is required to increase beyond a certain threshold; this is an example of a convex constraint. In general, defining actionability is a challenge: different settings require heterogeneous definitions of this criteria.

Our framework caters for a broad family of criteria: actionability can be expressed as any set of convex constraints. Algorithmically, such constraints can be appended to the feasibility checks in our decomposition algorithm (PICE).

In such a way, the decomposition will only return polyhedra that intersect actionable region (a condition that is appended at every feasibility check in Algorithm 2).

In other words, we restrict the algorithm's search space to counterfactuals within the actionability constraints. For a convex region of the input space $\mathcal{B}$, we add the constraint $x \in \mathcal{B}$ to Equation 1. Formally, the *plausible* counterfactual is defined as:

$$x^C_{\text{actionable}} = \underset{z \in \mathbb{R}^{n_0} \cap \mathcal{B}, \mathcal{N}(x) \neq \mathcal{N}(z)}{\text{argmin}} ||x - z||_2. \tag{3}$$

We note that for bounded regions with sufficiently small volumes, adding constraints reduces the computation time of the decomposition algorithm, since fewer polytopes need to be computed.

Sparsity through Model Truncation A sparse counterfactual is one where a small number of features have been changed with respect to the original query point. This count is measured by the L_0 norm: the number of values in the query vectors that have changed in the counterfactuals.

Sparse counterfactuals are often simpler to comprehend due to their small dimensional representation, which reduces the information that is conveyed to users. Moreover, sparsity may encourage simpler recourse (Ustun, Spangher, and Liu 2019): for example, a mortgage applicant who is provided a recourse option to increase both savings by 10% and credit rating by 5 points for a successful reapplication may have a higher cost than an applicant only tasked with increasing savings by the same amount.

However, we note a tension between sparsity and distance. By definition, a minimal counterfactual is as close as it can be to a query, meaning that if we enforce sparsity (thereby decreasing the distance) we will need to increase other the distance in other features to counterbalance the decrease in the first feature *and* the final counterfactual will be farther away. In other words, the second applicant will necessarily need to increase savings by an amount that is necessarily larger than 10%. This means that the cost to the user is a nuanced function of the input features, which would also take into account user preferences. Therefore, modellers may want to investigate the cost-sparsity tradeoff in the particular context of their problem setting to determine which features should be selected for sparse counterfactual.

Our method caters for any such choice of feature subsets, whenever this choice has been provided by the modellers. Otherwise, we may achieve sparse desiderata by changing only the most impactful features to the model. These may be determined automatically using feature importance techniques. Once a choice of features is provided, we search for a counterfactual only in the specified directions.

The process we employ to achieve sparse counterfactuals is outlined in Algorithm 6: the user can either choose dimensions in which they would like to optimise, or let a feature importance measures automatically determine which features are should be changed. We decompose a lower-dimensional *truncated* model $\mathcal{N}_S(x_S)$ for $x_S \in \mathbb{R}^{|S|}$ living in a lower-dimensional space contained in $\mathbb{R}^{n_0}$ the input space. We find weights that equalise the two networks on the subspace: for a fixed query point x such that x_S represents varying all features $S \subset [n_0] = \{1, 2, 3, ..., n_0\}$, we want to find $W^{(1)}, b^{(1)}$ such that $\mathcal{N}_S(x_S) = \mathcal{N}(x + \sum_{i \in S} a_i e_i), \forall x_S \in \mathbb{R}^S, \forall a_i \in \mathbb{R}, i \in S$, where e_i are basis vectors in the direction of i.

In other words, we create a faithful surrogate which mirrors the behaviour of the model exactly everywhere in the subspace where we are searching for a counterfactual. Once we have transported the problem in this simpler setting, we run PICE, targeting minimality as we would for the original network.

Formally, the sparse counterfactuals are given by:

$$x^C_{\text{sparse}} = \underset{z \in \mathbb{R}^{|S|} \cap \mathcal{B}, \mathcal{N}(x) \neq \mathcal{N}_S(z_S)}{\text{argmin}} ||x - z||_2. \tag{4}$$

Notice, how this generally results in a reduction in the complexity of the linear programs that need to be solved to decompose the network, leading to efficiency gains.

Plausibility: Selecting Inhabited Polytopes Plausibility broadly refers to the "in-manifoldness" of the counterfactual point. Although there are many contrasting definitions, we correlate the plausibility of a counterfactual as the property of being in a polyhedron that contains at least κ other data points.

Formally, suppose $\mathcal{D} \subset \mathbb{R}^{n_0}$ is a dataset from which we pick a query x. For a neural network $\mathcal{N}$ with decomposition $(\Omega, \{\alpha_\omega, \beta_\omega\}_{\omega \in \Omega})$, we define $\Omega^{\mathcal{D}} = \{\omega : \exists z_1, z_2, ..., z_\kappa \in \mathcal{D}, z_1, z_2, ..., z_\kappa \in \omega\}$ to be the set of κ-inhabited polyhedra. These polyhedra can be easily computed: once an activation pattern is computed by looking at the model internals at inference, the set of points in the polyhedron is given by the feasible set of constraints in Definition 3.1, as defined here:

$$x^C_{\text{plausible}} = \underset{z \in \Omega^{\mathcal{D}}, \mathcal{N}(x) \neq \mathcal{N}(z)}{\text{argmin}} ||x - z||_2. \tag{5}$$

This algorithm is similar to the sampling method described before, where the key difference is in the polyhedra that are selected. This approach mirrors that taken by Carreira-Perpiñán and Hada (2023) in the context of decision trees, with the notion of *live regions*.

Robustness Whenever counterfactuals are used for recourse, users update their initial query accordingly and may do so noisily. Therefore, it is desirable that the points in the neighbourhood of a query remain valid counterfactuals. Given that our points are exactly on the decision boundary, a large proportion of points sampled around our query for most standard choices of distribution (e.g. normal, or uniform in a hyperbox around the counterfactual). This tension between minimality and robustness has been remarked by Sharma et al. (2022). To achieve robustness, we update our computed counterfactual post-hoc, by moving it in the direction of the counterfactual vector $x^C - x$, away from the query. The resulting point,

$$x^C_{\text{robust}} = r(x^C - x) + x^C, \tag{6}$$

is the updated counterfactual for a choice of positive scalar $r \in \mathbb{R}^+$, which controls the degree of robustness. In Appendix B, we show experimentally that PICE enjoys a low invalidation rate on this ray.

Notice how without this modification our minimal counterfactuals will be fragile to noisy recourse. This is due to minimality: as Sharma, Henderson, and Ghosh (2019) remark, there is a fundamental tension between minimality and robustness. If a counterfactual is minimal, failing to reach the counterfactual's condition would lead to certain invalidation of recourse. Therefore, the minimal counterfactual is thought thought of as minimal changes that are necessary to for recourse. This post-processing step helps users by providing users with a counterfactual that allows a small margin of error, whenever modellers deem that this is desirable.

5 Experiments

Our experimental evaluation validates our theory, showing that we find the closest counterfactuals, and investigates the extent to which we are able to target specific counterfactual desiderata. We provide empirical support to the successful achievement of the following properties: minimality, sparsity, robustness and plausibility. Moreover, we show that our decomposer is able to effectively achieve the global decomposition more efficiently on a range of architectures compared to Villani and Schoots (2023) in Appendix A.

We compute counterfactuals on four datasets Adult (Becker and Kohavi 1996), HELOC (Fair Isaac Corporation 2018), Lending Club (Lending Club Corporation 2024) and German Credit (Hofmann 1994), picking 1000 negative queries, whenever these are available. For each dataset, we train classifiers on ReLU architectures, with three layers and seven neurons per layer (see Appendix D for training details and evidence that the size of the architecture is sufficient for acceptable predictive accuracy).

First, we provide a general comparison of our methods against popular techniques in the counterfactual and adversarial robustness literature. We compare against adversarial methods since they aggressively target minimality from query and decision boundary, offering a tougher comparison. We then compare our targeted algorithms against counterfactual techniques that optimise for the same desiderata. All counterfactuals are stored as `Numpy` objects, models as `Pytorch` objects and additional details are provided in Supplementary Material for replicability of experiments. Complexity of each method is analysed in Appendix E.

Counterfactual Comparison

Minimality: to assess the ability to find points on the decision boundary, we compute counterfactuals insensitive to the categorical/continuous feature distinction. We do this for methods that do not intrinsically handle categorical features differently. In this context, we measure distance with L_2 norm. Then, we address the treatment of categorical variables by providing results on a post-hoc clipping of the counterfactuals (detailed results in Appendix B). Categorical variables are one hot encoded; we apply clipping by taking the maximum argument of the counterfactual vector $x^C - x$ and assign a value of 1 only to the maximum argument, 0 elsewhere, interpreted as the prescribed direction of strongest change. We measure distance by Heterogenous Eucledian Overlap Metric (HEOM) (Wilson and Martinez 1997): simple matching on categorical features, added to L_2 for continuous features, recording average HEOM only on valid counterfactuals in Table **??**, and providing other experiments in the Appendix B. All measures, except speed, are only computed on valid counterfactuals.

We compare our counterfactuals against CCHVAE (Pawelczyk, Broelemann, and Kasneci 2020) and Wachter (Wachter, Mittelstadt, and Russell 2017) from the counterfactual literature and FAB (Croce and Hein 2020), Carlini-Wagner (Carlini and Wagner 2017), and Projected Gradient Descent method (Madry et al. 2017) from the adversarial robustness literature. These methods have been selected since they target minimality. In particular, adversarial robustness techniques can present an aggressive benchmark against which to compare since they strongly prioritise distance to query (Madry et al. 2017). We find these techniques on the Carla-Recourse (Pawelczyk et al. 2021) and Advertorch libraries (Ding, Wang, and Jin 2019). *Boundary Distance* in Table **??**, is a measure of distance from the decision boundary computed by finding the L_2 distance between the output and $[0.5, 0.5]$ point. In experiments, we interpret this measure as being optimised as it approaches zero.

Results show that our method is able to find points very close to the query instances, shown by the small average boundary and L_2 distances across all datasets in Table **??**. We achieve state-of-the-art against both best-performing counterfactual techniques and adversarial attacks in terms of distance from query, validity and latent distance to the decision boundary, while maintaining acceptable computation time. Our (Fast) variant improves on time, while maintaining competitive minimality metrics in comparison to counterfactual techniques.

Sparsity: for techniques targeting sparsity, we compute all previously mentioned metrics, which remain relevant, as well as the L_0 norm. This counts the number of features that have changed in the counterfactual, compared to the query. In Table **??**, we demonstrate how our PICE (Sparse) variant successfully finds minimal counterfactuals, while improving on the L_0 norm against PICE, our main method. This comes at the cost of validity, since the classifier does not attain multiple labels in all subspaces it explores, and an increased distance to query in the input space. We also compare against popular sparsity targeting methods in Appendix B: COGS (Virgolin and Fracaros 2023) and NICE (Brughmans, Leyman, and Martens 2023).

Plausibilily: to measure plausibility (Poyiadzi et al. 2020), we use Isolation Forest (IF) (Liu, Ting, and Zhou 2008) and Local Outlier Factor (LOF) (Cheng, Zou, and Dong 2019), and use percentage of in-liers as our metrics to assess the plausibility of counterfactuals. In both cases, we use the sci-kit learn implementations with default settings, except for the number of neighbours parameter, which we set to 5% of the dataset. For the lending dataset, we select the first 50,000 points to train the Local Outlier Factor for

METHOD	SPEED (S)	VALIDITY	BOUNDARY DISTANCE	L_2	HEOM
ADULT					
WACHTER	2.843×10^{-1}	100%	1.7×10^{-2}	6.873×10^{-1}	7.01
CCHVAE	1.967×10^{-1}	99.90%	6.2×10^{-2}	4.422×10^{0}	7.09
PICE	6.351×10^{1}	100%	$\mathbf{5.3 \times 10^{-13}}$	$\mathbf{6.424 \times 10^{-2}}$	**6.76**
PICE (FAST)	7.089×10^{-1}	100%	4.8×10^{-1}	5.068×10^{-1}	6.84
PICE (PLAUSIBLE)	5.047×10^{-1}	100%	$\mathbf{5.3 \times 10^{-13}}$	$\mathbf{6.424 \times 10^{-2}}$	**6.76**
LINFPGD	$\mathbf{1.541 \times 10^{-4}}$	88.80%	4.5×10^{-1}	2.630×10^{0}	7.40
FAB	6.842×10^{-4}	11.20%	7.8×10^{-5}	1.275×10^{-1}	7.02
CW	1.858×10^{-1}	87.80%	9.7×10^{-8}	3.347×10^{-1}	7.19
LENDING					
WACHTER	8.629×10^{-3}	100%	1.2×10^{-2}	8.733×10^{-2}	4.58
CCHVAE	9.264×10^{-2}	99.10%	4.0×10^{-3}	4.947×10^{0}	6.29
PICE	4.183×10^{1}	100%	$\mathbf{3.1 \times 10^{-15}}$	$\mathbf{1.452 \times 10^{-2}}$	**4.48**
PICE (FAST)	6.661×10^{-1}	100%	3.1×10^{-4}	4.528×10^{0}	5.23
PICE (PLAUSIBLE)	4.848×10^{-1}	100%	$\mathbf{3.1 \times 10^{-15}}$	$\mathbf{1.452 \times 10^{-2}}$	**4.48**
LINFPGD	$\mathbf{1.955 \times 10^{-4}}$	86.90%	3.0×10^{-2}	1.931×10^{0}	5.02
FAB	1.027×10^{-3}	17.80%	7.6×10^{-6}	3.608×10^{-2}	5.01
CW	9.515×10^{-2}	81.90%	5.2×10^{-10}	2.124×10^{-2}	5.02

Table 1: Comparison of Methods for Computing Counterfactuals (full experiments are provided in B, standard deviations are provided in Appendix B). All measures, except speed, are only computed on valid counterfactuals.

METHOD	SPEED (S)	VALIDITY	BOUNDARY DISTANCE	L_2	L_0
LENDING					
COGS	7.182×10^{-1}	100%	2.9×10^{-2}	3.860×10^{-1}	3.394
FACE	$\mathbf{1.343 \times 10^{1}}$	100%	5.4×10^{-3}	8.242×10^{0}	11.66
NICE	1.531×10^{-1}	100%	3.4×10^{-3}	3.093×10^{0}	7.786
PICE	1.399×10^{1}	100%	3.1×10^{-15}	$\mathbf{1.452 \times 10^{-2}}$	64.04
PICE (SPARSE)	8.937×10^{0}	86.80%	$\mathbf{2.9 \times 10^{-15}}$	2.132×10^{0}	**3.029**

Table 2: Demonstrating the ability of the sparse variant of PICE to find sparser counterfactuals. All measures, except speed, are only computed on valid counterfactuals.

tractability. We use FACE (Poyiadzi et al. 2020) as a baseline, which will provide a strong comparison on IS and LOF since it selects points in the dataset as counterfactuals.

Table **??** reports IF and LOF scores averaged across counterfactuals first, and across datasets afterwards. Disaggregated measures can be found in Appendix B. Interestingly, we find that PICE (without any additional desiderata enforcement) scores well in plausibility. Indeed, we find that the PICE and its plausible variant often converge to the same counterfactual point, suggesting that the closest decision boundary is often at a face or in an inhabited polyhedron. Whenever they disagree, PICE (Plausible) slightly improves the plausibility.

Robustness: we follow the experimental evaluation from Pawelczyk et al. (2022) to asses the robustness of our newly generated counterfactuals. This is usually verified by sampling points near the counterfactual according to some distribution. The choice of distribution varies (Sharma et al. 2022; Pawelczyk et al. 2022; Dominguez-Olmedo, Karimi, and Schölkopf 2022; Guidotti 2022; Virgolin and Fracaros 2023); we follow (Pawelczyk, Broelemann, and Kasneci 2020) in defining a hyperbox around the query from which we sample uniformly, but only in the direction of the counterfactual. We sample 1000 points in hyperboxes of varying size ($\epsilon = [0.01, 0.05, 0.1, 0.5, 1]$).

Table 4 reports the robustness for $\epsilon = 0.01$, and in the Appendix B we report full experimental evaluations. The PICE (Robust) presents a significant improvement in robustness, compared to PICE. It also performs well compared to other popular counterfactual methods, including COGS which targets specifically robustness. PICE (Fast) also performs well on robustness: this is because it generally may find points in polyhedra that are not adjacent to nor intersecting with the decision boundary.

6 Conclusion and Future Work

We found minimal counterfactuals for CPWL neural networks and developed methods to target popular desiderata. In particular, we provide theoretical guarantees that PICE converges to the closest points to the query with a different label. By leveraging the polyhedral geometry of the neural network, we provide variants that target popular desiderata in the counterfactual literature. Our experimental evaluation highlights clear advantages against existing methods in the pursuit of minimality, and improvements across all desiderata. In this way, we have shown how leveraging the functional form of the network can provide faithful explanations (Guidotti 2022) with guaranteed optimality.

METHOD	IF	LOF
CCHVAE	1.0000	0.9722
FACE	0.9594	0.9947
FAB	0.9276	0.9668
NICE	0.9271	0.9617
WACHTER	0.9074	0.9690
CW	0.9048	0.9870
PICE (PLAUSIBLE)	0.8915	0.9550
PICE	0.8905	0.9550
COGS	0.8437	0.8908
PICE (SPARSE)	0.8314	0.6726
PICE (FAST)	0.6313	0.6985
LINFPGD	0.2158	0.7294

Table 3: Comparison of Plausibility for counterfactuals, showing average percentage of inliers according to Isolation Forest (IS) and Local Outlier Factor (LOF) across datasets. Breakdown across datasets can be found in Table 8

Method	$\epsilon = 0.01$ **Robustness**	
	Adult	**Heloc**
CCHVAE	$95.8 \pm 11.0\%$	$79.6 \pm 17.3\%$
COGS	$100 \pm 0.00\%$	$100 \pm 0.705\%$
CW	$49.8 \pm 1.49\%$	$51.7 \pm 1.05\%$
FAB	$60.7 \pm 11.4\%$	$54.8 \pm 3.03\%$
FACE	$99.4 \pm 4.70\%$	$99.8 \pm 2.52\%$
LINFPGD	$100 \pm 0.553\%$	$100 \pm 0.00\%$
NICE	$100 \pm 0.00\%$	$94.4 \pm 12.3\%$
PICE	$50.1 \pm 0.554\%$	$51.7 \pm 0.00\%$
PICE (Fast)	$99.9 \pm 1.64\%$	$88.1 \pm 21.1\%$
PICE (Robust)	$98.8 \pm 5.85\%$	$98.5 \pm 6.24\%$
Wachter	$85.8 \pm 15.9\%$	$94.5 \pm 11.0\%$

Table 4: Robustness: measured by average validty rate for different hyperbox sizes, with standard deviations. Further experiments can be found in Table 10

The explainability of machine learning models, especially neural networks which are generally deemed as black-boxes, is imperative for developing responsible AI. Our work can not only be used extensively to help developers of continuous piecewise neural networks understand their models, but also help end-users subject to such models' decisions by providing them recourse recommendations. This work is a step towards the class of more interpretable models which can also provide substantial predictive power, and encourages future work in this space. The impact of our work is to provide greater insight into model behaviour for piecewise linear networks, which is expected to enhance safety and trustworthiness of these popular AI models, while providing benchmarks against which future counterfactual explanation methods can be compared.

Future directions of research would address some limitations of our work. Firstly, as discussed in Appendix E, improving the computational complexity of the main algorithm or tightening the guarantees of the fast variant would empower the computation of minimal counterfactuals in increasingly large networks. Secondly, our work is focused on ReLU-based architectures: while a generalisation to general piece-wise linear networks is directly achievable, a goal of future research is to adapt the algorithm to other activation functions (such as sigmoids and tanh).

Decision boundaries contain information about the model's vulnerability, its reasoning and complexity (Karimi, Derr, and Tang 2019; Karimi and Tang 2020) and can be used to adversarial attacks (Madry et al. 2017), measuring the expressivity of the network (Corlay et al. 2019), its generalisation at inference (Li, Ding, and Gao 2018), and the distance between a point and its closest point on the boundary can be used to model uncertainty, assess the robustness of the model (Lan, Brückner, and Lomuscio 2023) and explanations (Mohammadi et al. 2021). A corollary of our method is the ability to find points that are (a) exactly on the decision boundary and (b) have minimal distance. This has consequences in all of the above explorations, enabling better quantification of uncertainty, robustness of models and retraining. Moreover, exactly minimal counterfactuals enable a deeper exploration of robustness of counterfactual explanations: how quickly and dramatically does a counterfactual change if we change the query slightly? This line of inquiry is important to recourse, where it may be desirable to require of candidates with similar features to be treated similarly and, in particular, be given similar recourse options.

PICE enables new insight into piecewise linear neural networks through efficient and tailored explanations. In future work, we will look into using these models as surrogate models for less interpretable models or black-box models to explain their predictions. Moreover, we envision the utility of PICE to go beyond the computation of counterfactuals: it can be used as a benchmark for minimality, to explore properties of the decision boundary, and in adversarial training. These are promising directions, which can provide insight in the behaviour of networks and may lead to improvements in current methodologies.

Acknowledgements

(1) This paper/presentation was prepared for informational purposes in part by the CDAO group of JPMorgan Chase & Co and its affiliates ("J.P. Morgan") and is not a product of the Research Department of J.P. Morgan. J.P. Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful. (2) This work was supported by UK Research and Innovation, in the UKRI Centre for Doctoral Training in Safe and Trusted Artificial Intelligence (www.safeandtrustedai.org).

References

Arora, S.; Cohen, N.; and Hazan, E. 2018. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, 244–253. PMLR.

Balestriero, R.; and LeCun, Y. 2023. Fast and Exact Enumeration of Deep Networks Partitions Regions. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. IEEE.

Becker, B.; and Kohavi, R. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

Berzins, A. 2023. Polyhedral Complex Extraction from ReLU Networks using Edge Subdivision. *arXiv preprint arXiv:2306.07212*.

Brughmans, D.; Leyman, P.; and Martens, D. 2023. Nice: an algorithm for nearest instance counterfactual explanations. *Data Mining and Knowledge Discovery*, 1–39.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, 39–57. Ieee.

Carreira-Perpiñán, M. Á.; and Hada, S. S. 2023. Very fast, approximate counterfactual explanations for decision forests. *arXiv preprint arXiv:2303.02883*.

Cheng, Z.; Zou, C.; and Dong, J. 2019. Outlier detection using isolation forest and local outlier factor. In *Proceedings of the conference on research in adaptive and convergent systems*, 161–168.

Corlay, V.; Boutros, J. J.; Ciblat, P.; and Brunel, L. 2019. A lattice-based approach to the expressivity of deep ReLU neural networks. *arXiv preprint arXiv:1902.11294*.

Croce, F.; and Hein, M. 2020. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, 2196–2205. PMLR.

Ding, G. W.; Wang, L.; and Jin, X. 2019. AdverTorch v0. 1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*.

Dominguez-Olmedo, R.; Karimi, A. H.; and Schölkopf, B. 2022. On the adversarial robustness of causal algorithmic recourse. In *International Conference on Machine Learning*, 5324–5342. PMLR.

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

Eldan, R.; and Shamir, O. 2016. The power of depth for feedforward neural networks. In *Conference on learning theory*, 907–940. PMLR.

Fair Isaac Corporation. 2018. HELOC Dataset. FICO Explainable Machine Learning Challenge.

Grigsby, E.; Lindsey, K.; and Rolnick, D. 2023. Hidden symmetries of ReLU networks. In *International Conference on Machine Learning*, 11734–11760. PMLR.

Grigsby, J. E.; and Lindsey, K. 2022. On transversality of bent hyperplane arrangements and the topological expressiveness of ReLU neural networks. *SIAM Journal on Applied Algebra and Geometry*, 6(2): 216–242.

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

Hanin, B.; and Rolnick, D. 2019. Deep relu networks have surprisingly few activation patterns. *Advances in neural information processing systems*, 32.

Hanin, B.; and Sellke, M. 2017. Approximating continuous functions by relu nets of minimal width. *arXiv preprint arXiv:1710.11278*.

Hofmann, H. 1994. Statlog (German Credit Data). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77.

Humayun, A. I.; Balestriero, R.; and Baraniuk, R. 2022. Exact visualization of deep neural network geometry and decision boundary. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*.

Hutson, M. 2018. Artificial intelligence faces reproducibility crisis.

Karimi, A.-H.; Schölkopf, B.; and Valera, I. 2021. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, 353–362.

Karimi, H.; Derr, T.; and Tang, J. 2019. Characterizing the decision boundary of deep neural networks. *arXiv preprint arXiv:1912.11460*.

Karimi, H.; and Tang, J. 2020. Decision boundary of deep neural networks: Challenges and opportunities. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 919–920.

Lan, J.; Brückner, B.; and Lomuscio, A. 2023. A semidefinite relaxation based branch-and-bound method for tight neural network verification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 14946–14954.

Lei, J.; and Jia, K. 2020. Analytic marching: An analytic meshing solution from deep implicit surface networks. In *International Conference on Machine Learning*, 5789–5798. PMLR.

Lending Club Corporation. 2024. Lending Club Loan Data. Historical data on loans issued including performance metrics.

Li, Y.; Ding, L.; and Gao, X. 2018. On the decision boundary of deep neural networks. *arXiv preprint arXiv:1808.05385*.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*, 413–422. IEEE.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267: 1–38.

Mohammadi, K.; Karimi, A.-H.; Barthe, G.; and Valera, I. 2021. Scaling guarantees for nearest counterfactual explanations. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 177–187.

Montufar, G. F.; Pascanu, R.; Cho, K.; and Bengio, Y. 2014. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27.

Mothilal, R. K.; Sharma, A.; and Tan, C. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, 607–617.

O'Neill, B. 2023. Three distributions in the extended occupancy problem. *Methodology and Computing in Applied Probability*, 25(4): 84.

Pawelczyk, M.; Bielawski, S.; Heuvel, J. v. d.; Richter, T.; and Kasneci, G. 2021. Carla: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *arXiv preprint arXiv:2108.00783*.

Pawelczyk, M.; Broelemann, K.; and Kasneci, G. 2020. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of the web conference 2020*, 3126–3132.

Pawelczyk, M.; Datta, T.; van-den Heuvel, J.; Kasneci, G.; and Lakkaraju, H. 2022. Probabilistically robust recourse: Navigating the trade-offs between costs and robustness in algorithmic recourse. *arXiv preprint arXiv:2203.06768*.

Poyiadzi, R.; Sokol, K.; Santos-Rodriguez, R.; De Bie, T.; and Flach, P. 2020. FACE: feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 344–350.

Raghu, M.; Poole, B.; Kleinberg, J.; Ganguli, S.; and Sohl-Dickstein, J. 2017. On the expressive power of deep neural networks. In *international conference on machine learning*, 2847–2854. PMLR.

Rolnick, D.; and Kording, K. 2020. Reverse-engineering deep relu networks. In *International Conference on Machine Learning*, 8178–8187. PMLR.

Schleich, M.; Geng, Z.; Zhang, Y.; and Suciu, D. 2021. Geco: Quality counterfactual explanations in real time. *arXiv preprint arXiv:2101.01292*.

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

Sharma, S.; Gee, A. H.; Henderson, J.; and Ghosh, J. 2022. FASTER-CE: Fast, Sparse, Transparent, and Robust Counterfactual Explanations. *arXiv preprint arXiv:2210.06578*.

Sharma, S.; Henderson, J.; and Ghosh, J. 2019. Certifai: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. *arXiv preprint arXiv:1905.07857*.

Sharma, S.; Sharma, S.; and Athaiya, A. 2017. Activation functions in neural networks. *Towards Data Sci*, 6(12): 310–316.

Sotoudeh, M.; and Thakur, A. V. 2019. Computing linear restrictions of neural networks. *Advances in Neural Information Processing Systems*, 32.

Sudjianto, A.; Knauth, W.; Singh, R.; Yang, Z.; and Zhang, A. 2020. Unwrapping the black box of deep relu networks: Interpretability, diagnostics, and simplification. *arXiv preprint arXiv:2011.04041*.

Szandała, T. 2021. Review and comparison of commonly used activation functions for deep neural networks. *Bio-inspired neurocomputing*, 203–224.

Tran, K. H.; Ghazimatin, A.; and Saha Roy, R. 2021. Counterfactual explanations for neural recommenders. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1627–1631.

Ustun, B.; Spangher, A.; and Liu, Y. 2019. Actionable recourse in linear classification. In *Proceedings of the conference on fairness, accountability, and transparency*, 10–19.

Van Looveren, A.; and Klaise, J. 2021. Interpretable counterfactual explanations guided by prototypes. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 650–665. Springer.

Verma, S.; Boonsanong, V.; Hoang, M.; Hines, K. E.; Dickerson, J. P.; and Shah, C. 2020. Counterfactual explanations and algorithmic recourses for machine learning: A review. *arXiv preprint arXiv:2010.10596*.

Villani, M. J.; and Schoots, N. 2023. Any Deep ReLU Network is Shallow. *arXiv preprint arXiv:2306.11827*.

Virgolin, M.; and Fracaros, S. 2023. On the robustness of sparse counterfactual explanations to adverse perturbations. *Artificial Intelligence*, 316: 103840.

Wachter, S.; Mittelstadt, B.; and Russell, C. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.*, 31: 841.

Wang, Y. 2022. Estimation and comparison of linear regions for relu networks. IJCAI.

Wilson, D. R.; and Martinez, T. R. 1997. Improved heterogeneous distance functions. *Journal of artificial intelligence research*, 6: 1–34.

Wu, A.; Sadraddini, S.; and Tedrake, R. 2020. The nearest polytope problem: Algorithms and application to controlling hybrid systems. In *2020 American Control Conference (ACC)*, 1815–1822. IEEE.

Yang, W.; Li, J.; Xiong, C.; and Hoi, S. C. 2022. Mace: An efficient model-agnostic framework for counterfactual explanation. *arXiv preprint arXiv:2205.15540*.