The Minimal Feature Removal Problem in Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

We present the *minimal feature removal problem* for neural networks, a combinatorial problem which has interesting potential applications for improving interpretability and robustness of neural network predictions. For a given input to a trained neural network, our aim is to compute a smallest set of input features so that the model prediction changes when these features are disregarded by setting them to a given uninformative baseline value. We show that computing such minimal subsets of features is computationally intractable for fully-connected neural networks with ReLU nonlinearities. We show, however, that the problem becomes solvable in polynomial time by a greedy algorithm for monotonic networks. We then show that our tractability result extends seamlessly to more advanced neural network architectures such as convolutional and graph neural networks under suitable monotonicity assumptions.

1 Introduction

Deep Neural Networks have experienced unprecedented success in areas such as image analysis, natural language processing, speech recognition, and data science, with systems outperforming humans in a wide range of tasks Krizhevsky et al. (2012); Hannun et al. (2014); LeCun et al. (2015); Schmidhuber (2015); Silver et al. (2016). As the use of neural models becomes widespread in complex applications, however, task performance is no longer the only driver of system design, and criteria such as safety, fairness, and robustness have gained significant prominence in recent years Kazim & Koshiyama (2021).

Improving model interpretability is an important step towards fulfilling these criteria: if models can explain their predictions, it becomes easier to ensure that predictions based on them are safe and fair. This is, however, notoriously challenging, as neural models are 'black boxes' where predictions rely on numeric calculations in high-dimension embedding spaces. A wealth of different explanation approaches have been proposed in recent years: *rule-based methods* generate explanations in the form of logic rules, which are inherently interpretable Cucala et al. (2022); Dhurandhar et al. (2018); *attribution-based methods* assign a score to input features quantifying their contribution to the prediction relative to a baseline Sundararajan et al. (2017); Sundararajan & Najmi (2020); Ancona et al. (2018); *example-based methods* explain predictions by retrieving training examples that are most similar to the given input Koh & Liang (2017); Li et al. (2018); and *perturbation-based methods* generate corrections to an input causing the model to change its output Zhang et al. (2018); Goyal et al. (2019); Lucic et al. (2022); Bajaj et al. (2021). Evaluating and improving the robustness of neural predictions is also an important and challenging problem Carlini & Wagner (2017); Hendrycks & Dietterich (2019). Indeed, neural networks are highly sensitive to a wide range of adversarial attacks, which creates significant vulnerabilities and safety risks when these models are applied in mission-critical applications Szegedy et al. (2014); Katz et al. (2017). Explainability and robustness properties often go hand-in-hand; for instance, attribution-based explanation methods are vulnerable to adversarial attacks since small changes in the input can lead to significant changes on the assigned attribution scores Ghorbani et al. (2019); Dombrowski et al. (2019); Yeh et al. (2019).

In this paper we introduce the *minimal feature removal problem*: a combinatorial problem with interesting potential applications for improving the interpretability and robustness of neural model predictions. Given an (application dependent) baseline representing absence of information and an input feature vector to a trained network, our aim is to compute a smallest set S of input features so that the prediction changes when these features are set to their corresponding baseline value. The minimality requirement ensures that S can be interpreted as a form of explanation which captures the 'essence' of the corresponding prediction by identifying genuinely irrelevant features that can be disregarded without an observable effect. Furthermore, the number of features in S provides an intuitive measure for the robustness of the prediction since our definition formally guarantees that disregarding any smaller number of input features cannot change the prediction. Finally, S may also be interpreted as a form of adversarial attack and, in particular, a type of ℓ_0-perturbation Kotyan & Vargas (2022), where the goal is to change the prediction by identifying changes to a discrete set of input features; indeed, as discussed in Li et al. (2022); Hosseini et al. (2019), ℓ_0-perturbations cannot be seen as a continuous optimisation problem and require a combinatorial approach.

Our approach effectively combines ideas from attribution-based and perturbation-based methods. As in attribution-based explanation methods, we introduce an uninformative baseline as reference and, similarly to perturbation-based methods for explanation and adversarial attack, our identified set of features constitutes a minimal input correction with an observable effect. Our notion of feature removal is, however, rather different from related perturbation-based approaches Zhang et al. (2018); Lucic et al. (2022); Bajaj et al. (2021); Fong & Vedaldi (2017); Goyal et al. (2019) in that the aim is to 'toggle off' features using the baseline rather than identifying arbitrary value changes.

Our contributions are as follows. After introducing the notion of minimal feature removal, we focus on fully-connected networks as a starting point and show that computing minimal feature removal sets is an intractable problem, even with ReLU activations. If, however, the matrix weights in the network are non-negative and the non-linear activations of the network are continuous, differentiable almost everywhere and monotonic (as is the case with standard activations) we can show that minimal feature removal sets are computable in polynomial time by a greedy algorithm. To show correctness of our greedy algorithm, we exploit the theoretical properties of the *integrated gradients* method Sundararajan et al. (2017), thus establishing an interesting connection between our approach and the theory of attribution methods; our algorithm, however, does not rely on the application of any such attribution method, and only requires the ability to run the neural network as a black box. We then further generalise our results to more advanced neural architectures such as convolutional neural networks (CNNs) Goodfellow et al. (2016); Krizhevsky et al. (2012) and graph neural networks (GNNs) Gori et al. (2005); Kipf & Welling (2017) and show that the problem remains tractable under suitable generalisations of the monotonicity requirement.

Although our intractability result is applicable to a many network architectures used in practice in applications such as image analysis, our greedy algorithm can still be exploited for a wide range of learning tasks where the underpinning function is monotonic Marques-Silva et al. (2021); Cano et al. (2019); Cucala et al. (2022).

When efficiently computable, minimal feature removal sets may provide interesting insights on the interpretability and the robustness of predictions. Thus, our algorithm can become a valuable tool for both designers and users of neural networks.

2 Preliminaries

Notation. We let bold-face lowercase letters denote real-valued vectors, and typically use $\mathbf{x}$ for input feature vectors and $\mathbf{x}'$ for baselines. Given vector $\mathbf{x}$, we use x_i to denote its i-th component. Given vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$ and a subset $S \subseteq \{1, ..., n\}$ of their components, we denote with $\mathbf{x}^{S|\mathbf{x}'}$ the vector obtained from $\mathbf{x}$ by setting each component x_i with $i \in S$ to x'_i. We use bold-face capital letters to denote real-valued matrices and, given matrix $\mathbf{M}$, we denote its (i, j) component as $M_{i,j}$. Finally, given a function $f : \mathbb{R}^n \mapsto \mathbb{R}$, we denote with $(\nabla f)_i$ the i-th component of its gradient.

Fully-Connected Neural Networks. A fully-connected neural network (FCN) with $L \geq 1$ layers and input dimension n is a tuple $\mathcal{N} = \langle \{\mathbf{W}^\ell\}_{1 \leq \ell \leq L}, \{\mathbf{b}^\ell\}_{1 \leq \ell \leq L}, \{\sigma^\ell\}_{1 \leq \ell \leq L} \rangle$. For each layer $\ell \in \{1, \ldots, L\}$, the integer $d_\ell \in \mathbb{N}$ is the *width* of layer ℓ and we require $d_L = 1$ and define $d_0 = n$; matrix $\mathbf{W}^\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ is a *weight matrix*; vector $\mathbf{b}^\ell \in \mathbb{R}^{d_\ell}$ is a *bias vector*; and $\sigma^\ell : \mathbb{R} \mapsto \mathbb{R}$ is a polytime-computable *activation function* applied component-wise to vectors.

The application of network $\mathcal{N}$ to an input feature vector $\mathbf{x} \in \mathbb{R}^n$ generates a sequence $\mathbf{x}^1, \ldots, \mathbf{x}^L$ of vectors defined as $\mathbf{x}^\ell = \sigma^\ell(\mathbf{h}^\ell)$, where $\mathbf{x}^0 = \mathbf{x}$ and $\mathbf{h}^\ell = \mathbf{W}^\ell \cdot \mathbf{x}^{\ell-1} + \mathbf{b}^\ell$. The result $\mathcal{N}(\mathbf{x})$ of applying $\mathcal{N}$ to $\mathbf{x}$ is the scalar $\mathbf{x}^L$. Thus, the neural network realises a function $\mathcal{N} : \mathbb{R}^n \mapsto \mathbb{R}$.

When for each $\ell \in \{1, \ldots, L\}$, σ^ℓ is the rectified linear unit (ReLU), i.e. $\sigma^\ell(x) = \max(0, x)$, we say that $\mathcal{N}$ is a ReLU FCN.

We consider the application of neural networks to classification problems. To avoid trivialities, we focus on classifying an input in two classes based on a numeric threshold $t \in \mathbb{R}$. In this setting, the *prediction* of a network $\mathcal{N}$ on input $\mathbf{x}$ is given by the result of comparing $\mathcal{N}(\mathbf{x})$ to the threshold t.

3 Background on Attribution-based Methods

Attribution methods Sundararajan et al. (2017); Sundararajan & Najmi (2020); Shapley (1953) are a family of explanation techniques which, given as input function $f : \mathbb{R}^n \mapsto \mathbb{R}$, a vector $\mathbf{x} \in \mathbb{R}^n$ and a baseline vector $\mathbf{x}' \in \mathbb{R}^n$, assign a numerical score or contribution $C_i^f(\mathbf{x}, \mathbf{x}')$ to each component $i \in \{1, ..., n\}$. Attribution methods are often designed to fulfil some (or all) of the following axioms for all functions $\mathbb{R}^n \mapsto \mathbb{R}$ and vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, components $1 \leq i \leq n$ and coefficients $\lambda_1, \lambda_2 \in \mathbb{R}$.

- *Completeness:* $f(\mathbf{x}) - f(\mathbf{x}') = \sum_{j=1}^n C_j^f(\mathbf{x}, \mathbf{x}')$.
- *Zero-contribution:* $C_i^f(\mathbf{x}, \mathbf{x}') = 0$ whenever $f(\mathbf{y}) = f(y_1, ..., y_{i-1}, z, y_{i+1}, ...y_n)$ for each $\mathbf{y} \in \mathbb{R}^n$ and each $z \in \mathbb{R}$.
- *Symmetry:* $C_i^f(\mathbf{x}, \mathbf{x}') = C_j^f(\mathbf{x}, \mathbf{x}')$ if $x_i = x_j$, $x_i' = x_j'$ and $f(y_1, ..., y_i, ..., y_j, ...y_n) = f(y_1, ..., y_j, ..., y_i, ...y_n)$ for each $\mathbf{y} \in \mathbb{R}^n$.
- *Linearity:* $C_i^{\lambda_1 f_1 + \lambda_2 f_2}(\mathbf{x}, \mathbf{x}') = \lambda_1 C_i^{f_1}(\mathbf{x}, \mathbf{x}') + \lambda_2 C_i^{f_2}(\mathbf{x}, \mathbf{x}')$.

Completeness ensures that contributions add up to the change in value of the function. Zero-contribution ensures that arguments not influencing the value of the function are assigned 0 as contribution. Symmetry ensures that arguments playing a symmetric role are assigned the same contribution. Finally, linearity ensures that contributions for a function expressed as a linear combination of other functions can be computed as a linear combination of their contributions.

A wide range of attribution-based methods have been proposed. The *Shapley values* method Shapley (1953) is one of the most popular thanks to its nice properties. Calculating Shapley values is, however, intractable, which has motivated research on approximation techniques Ancona et al. (2019). Other popular attribution methods have been designed specifically for neural networks; these include *Layer-wise Relevance Propagation* Bach et al. (2015), *DeepLIFT* Shrikumar et al. (2017), *Deep Taylor decompositions* Montavon et al. (2017), and *Saliency Maps* Simonyan et al. (2014); Adebayo et al. (2018); Dabkowski & Gal (2017); Chang et al. (2017).

We will exploit the properties of *Integrated Gradients* Sundararajan et al. (2017); Aumann & Shapley (1974), which is an attribution method applicable to continuous functions that are differentiable almost everywhere. The contribution of each argument i of a such function f for input vector $\mathbf{x}$ and baseline vector $\mathbf{x}'$ is defined as follows:

$$C_i^f(\mathbf{x}, \mathbf{x}') := (x_i - x_i') \int_0^1 (\nabla f)_i \left(\mathbf{x}' + \tau(\mathbf{x} - \mathbf{x}')\right) \mathrm{d}\tau. \tag{1}$$

Integrated gradients is the only path-based attribution method satisfying all of the aforementioned axioms Friedman (2004). Furthermore, it is well-suited for functions realised by neural networks, which typically satisfy its continuity and differentiability requirements.

4 Minimal Feature Removal Sets

In this section, we first present our notion of a feature removal set and then show that computing a minimal such set is an intractable problem for fully-connected neural networks.

Given a trained FCN $\mathcal{N}$ providing a prediction on an input feature vector $\mathbf{x}$, we aim at identifying a smallest set of 'most relevant' features for the prediction. Similarly to attribution-based methods, we consider an uninformative baseline $\mathbf{x}'$ for comparison where the use of a suitable baseline can be exploited to 'toggle off' or disregard certain features; however, while attribution-based methods determine feature relevance by assigning a numeric value to each component of the input $\mathbf{x}$, our approach is based in the intuition that the most relevant features are those that result in a change of prediction when set to the baseline value.

Definition 1. *Let $\mathcal{N}$ be a FCN with L layers and input dimension n, let t be a numeric threshold, let $\mathbf{x} \in \mathbb{R}^n$ be such that $\mathcal{N}(\mathbf{x}) > t$, and let $\mathbf{x}' \in \mathbb{R}^n$ be a baseline vector. A* feature removal set *for $\mathcal{N}(\mathbf{x}) > t$ relative to $\mathbf{x}'$ is a subset $S \subseteq \{1, \ldots, n\}$ satisfying $\mathcal{N}(\mathbf{x}^{S|\mathbf{x}'}) \leq t$. The size $|S|$ of S is the number of elements it contains. Furthermore, we say S is* minimal *if no $S' \subset \{1, \ldots, n\}$ such that $|S'| < |S|$ is a feature removal set for $\mathcal{N}(\mathbf{x}) > t$ relative to $\mathbf{x}'$.*

Note that this definition implies that a feature removal set S cannot be the empty set.

Although different baselines may lead to different feature removal sets for the same prediction, a natural baseline independent from the input can often be identified Sundararajan et al. (2017).

We next show that the decision version of our problem, even restricted to ReLU FCNs, is NP-complete. Thus, to identify a minimal feature removal set, one may need to consider exponentially-many combinations of input features.

Theorem 1. *The following problem is NP-complete: given as input vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, a ReLU FCN $\mathcal{N}$ of input dimension n, a numeric threshold t such that $\mathcal{N}(\mathbf{x}) > t$, and $1 \leq k \leq n$, decide whether there exists a feature removal set for $\mathcal{N}(\mathbf{x}) > t$ relative to $\mathbf{x}'$ of size at most k.*

Proof. We show NP-hardness using a reduction from SUBSET-SUM, which is the problem of checking, given as input integers $a_1, ..., a_m$ and ξ, whether there exists $S \subseteq \{1, \ldots, m\}$ such that $\sum_{i \in S} a_i = \xi$. In the context of this proof, let us denote with $\mathbf{0}_m$ the m-dimensional column null vector and with $\mathbf{1}_m$ the m-dimensional column vector with all components equal to 1.

We map an instance $a_1, ..., a_m, \xi$ of SUBSET-SUM to an instance of our problem by setting $k = m$, $t = 0$, $\mathbf{x} = \mathbf{1}_{m+1}$, $\mathbf{x}' = \mathbf{0}_{m+1}$ and $\mathcal{N}$ the 2-layer ReLU network defined as given next. In the first layer, $\mathbf{W}^1$ is a $(3 \times m + 1)$ matrix with values $a_1, \ldots, a_m, 0$ in the first row, $-a_1, \ldots, -a_m, 0$ in the second row, and having $\mathbf{1}_{m+1}$ in the third row; furthermore, $\mathbf{b}^1_1 = -\sum_{i=1}^m a_i - \xi$, $\mathbf{b}^1_2 = \sum_{i=1}^m a_i + \xi$ and $\mathbf{b}^1_3 = -m$. In the second layer, $\mathbf{W}^2 = \mathbf{1}_3$, $\mathbf{b}^2 = 0$. With input $\mathbf{x} = \mathbf{1}_{m+1}$, we have that $h^1_1 = -\xi$, $h^1_2 = \xi$, and $h^1_3 = 1$ which yields that $h^2 \geq 1$. By construction, we thus have $\mathcal{N}(\mathbf{x}) > 0$, so each valid input to SUBSET-SUM is mapped to a valid input of our problem.

Assume there is a feature removal set S for $\mathcal{N}(\mathbf{x}) > 0$ relative to $\mathbf{x}'$ of size at most $k = m$. By definition, $\mathcal{N}(\mathbf{x}^{S|\mathbf{x}'}) \leq 0$. We claim that, by construction of $\mathcal{N}$, S is a solution to the corresponding subset sum instance. Indeed, we must have $h^2 \leq 0$ by property of the ReLU and by construction, $h^2 \geq 0$ (as a sum of ReLU activations), thus we must have $h^2 = 0$. This enforces that $h^1_1 \leq 0$ and $h^1_2 \leq 0$ by property of the ReLU (we also have $h^1_3 = 0$ since $S \neq \emptyset$). Since $h^1_1 = -h^1_2$, this yields $h^1_1 = h^1_2 = 0$. By construction, with input $\mathbf{x}^{S|\mathbf{x}'}$, we have $h^1_1 = \sum_{i \in \{1,\ldots,m\} - S} a_i - \sum_{i=1}^m a_i - \xi = -(\xi - \sum_{i \in S} a_i)$. It follows that S is a solution to $\sum_{i \in S} a_i = \xi$. For the converse, let S be a solution to SUBSET-SUM. We claim that S is also a solution to our problem. Indeed, $S \neq \emptyset$, thus with input $\mathbf{x}^{S|\mathbf{x}'}$, we have $h^1_3 = 0$. Furthermore, $\sum_{i \in S} a_i = \xi$ thus $h^1_1 = h^1_2 = 0$ which yields $h^2 = 0$ and therefore $\mathcal{N}(\mathbf{x}^{S|\mathbf{x}'}) \leq 0$.

Membership in NP follows since a set $S \subseteq \{1, \ldots, m\}$ of size at most k provides a certificate. In particular, S is a feature removal set if $\mathcal{N}(\mathbf{x}^{S|\mathbf{x}'}) \leq t$, which is verifiable in polynomial time. □

The proof of the theorem shows that intractability depends on the dimension of the feature and baseline vectors, but not on their concrete values. Neural networks, however, can have a large number of input features, which makes the computation of minimal feature removal sets problematic. Thus, our focus in the remainder of this paper will be on identifying additional requirements on the trained neural model ensuring that minimal feature removal sets can be computed in polynomial time.

5 Monotonic Neural Networks

A key observation in the proof of Theorem 1 is that the weights in the network have to take both positive and negative values to reduce to SUBSET-SUM. This observation suggests the following definition of a *monotonic* FCN.

Definition 2. *A fully-connected neural network* $\mathcal{N} = \langle \{\mathbf{W}^\ell\}_{1\le\ell\le L}, \{\mathbf{b}^\ell\}_{1\le\ell\le L}, \{\sigma^\ell\}_{1\le\ell\le L\}} \rangle$ *is* monotonic *if, for each* $1 \le \ell \le L$*, weights* $\mathbf{W}^\ell$ *are non-negative and function* σ^ℓ *is continuous everywhere, differentiable almost everywhere, and monotonic.*

In the remainder of this section we present our main result, which establishes that minimal feature removal sets can be computed in polynomial time for monotonic FCNS by means of a greedy algorithm. Thus, our intractability result in Theorem 1 may not apply to all FCNs used in practice.

5.1 Properties of Monotonic Neural Networks

Consider a trained FCN $\mathcal{N}$ where the weights and activation functions satisfy the requirements in Definition 2. The continuity and differentiability requirements ensure that the gradient of $\mathcal{N}$ can be computed for each input vector $\mathbf{x}$. In turn, as we show next, the monotonicity requirement ensures that each component of the gradient of $\mathcal{N}$ at $\mathbf{x}$ can be expressed as a sum where *(1)* the number of elements in the sum is fixed for $\mathcal{N}$ (i.e., it does not depend on $\mathbf{x}$) and it is the same for all components; and *(2)* each element of the sum consists of a product involving a value that depends on $\mathbf{x}$ but which is *always non-negative*, and two coefficients that do not depend on $\mathbf{x}$.

This key property of the gradient in monotonic FCNs allows us to exploit the theoretical properties of the integrated gradients attribution method. In particular, we can show that, by setting a component of the input vector $\mathbf{x}$ to the baseline, we are not altering the relative order of the integrated gradient attributions (computed as in equation 1) associated with the remaining components.

These properties, which are established by the following technical lemma, constitute the basis of our greedy algorithm for computing minimal feature removal sets.

Lemma 1. *For each monotonic FCN* $\mathcal{N}$ *with input dimension* n*, there exists a non-negative integer* M*, positive coefficients* $\{A_m\}_{1\le m\le M}$*, real coefficients* $\{B_i\}_{1\le i\le n}$*, and functions* $\{g_m\}_{1\le m\le M}$ *from* $\mathbb{R}^n$ *to* $\mathbb{R}_{\ge 0}$*, such that the following identities are satisfied for each* $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$ *and each* $i, j \in \{1, ..., n\}$*:*

$$(\nabla\mathcal{N})_i(\mathbf{x}) \quad = \quad B_i \sum_{m=1}^{M} A_m\, g_m(\mathbf{x}), \qquad \textit{and} \tag{2}$$

$$\mathcal{N}(\mathbf{x}^{\{i\}|\mathbf{x}'}) - \mathcal{N}(\mathbf{x}^{\{j\}|\mathbf{x}'}) \quad = \quad (B_i(x'_i - x_i) - B_j(x'_j - x_j)) \sum_{m=1}^{M} A_m \int_0^1 g_m(\mathbf{p}^{ij}(\tau))\mathrm{d}\tau \tag{3}$$

where $\mathbf{p}^{ij}(\tau) = \mathbf{x}^{\{j\}|\mathbf{x}'} + \tau(\mathbf{x}^{\{i\}|\mathbf{x}'} - \mathbf{x}^{\{j\}|\mathbf{x}'})$.

Proof. We show equation 2 by induction on the number L of layers in $\mathcal{N}$. If $L = 1$, then $\mathcal{N} = \langle \mathbf{W}, b, \sigma \rangle$ with $\mathbf{W}$ an n-dimensional vector, b a scalar and $\sigma : \mathbb{R} \mapsto \mathbb{R}$. Using the chain rule, we have $(\nabla\mathcal{N})_i(\mathbf{x}) = W_i \cdot (D\sigma)(\mathbf{W}\cdot\mathbf{x} + b)$, where $D\sigma$ is the derivative of σ in Euler's notation. This expression is of the form equation 2 with $M = 1$, $A_1 = 1 \ge 0$, $B_i = W_i$ if σ is monotonically increasing (dually, $B_i = -W_i$ if σ is monotonically decreasing), and $g_1(\mathbf{x}) = (D\sigma)(\mathbf{W}\cdot\mathbf{x} + b)$ if σ is monotonically increasing (dually, $g_1 = -(D\sigma)(\mathbf{W}\cdot\mathbf{x} + b)$ if σ is monotonically decreasing). In both cases, monotonicity of σ ensures that $g_1(\mathbf{x}) \ge 0$ for any $\mathbf{x}$.

For the inductive case, assume that equation 2 holds for every network with $L-1$ layers. The application of $\mathcal{N} = \langle \{\mathbf{W}^\ell\}_{1\le\ell\le L}, \{\mathbf{b}^\ell\}_{1\le\ell\le L}, \{\sigma^\ell\}_{1\le\ell\le L\}} \rangle$ with L layers to $\mathbf{x}$ is defined as $\sigma^L(h^L(\mathbf{x}))$. Thus, we can apply the chain rule and the definition of h^L to obtain the following identity:

$$(\nabla\mathcal{N})_i(\mathbf{x}) = (\nabla h^L)_i(\mathbf{x}) \cdot (D\sigma^L)(h^L(\mathbf{x})) = \sum_{j=1}^{d_{L-1}} W^L_j \cdot (\nabla\mathcal{N}^j)_i(\mathbf{x}) \cdot (D\sigma^L)(h^L(\mathbf{x})). \tag{4}$$

Here, $\mathcal{N}^j$ is specified by $\langle \{\mathbf{W}^\ell\}_{1\le\ell\le L-2}, \mathbf{W}^{L-1}_j, \{\mathbf{b}^\ell\}_{1\le\ell\le L-2}, b^{L-1}_j, \{\sigma^\ell\}_{1\le\ell\le L-1\}} \rangle$, where $\mathbf{W}^{L-1}_j$ and b^{L-1}_j are the j-th row of $\mathbf{W}^{L-1}$ and the j-th element of $\mathbf{b}^{L-1}$, respectively.

We apply the inductive hypothesis to obtain the value of the gradient for each $1 \le j \le d_L - 1$, which is given by $(\nabla\mathcal{N}^j)_i(\mathbf{x}) = B^j_i \ \sum_{m_j=1}^{M_j} A^j_{m_j}\ g^j_{m_j}(\mathbf{x})$. But now, we can replace the value

Algorithm 1

```
Input: monotonic FCN $\mathcal{N}$, vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$ and threshold $t \in \mathbb{R}$ such that $\mathcal{N}(\mathbf{x}) > t$.
for $1 \leq j \leq n$ do
    $c_j \leftarrow \mathcal{N}(\mathbf{x}^{\{j\}|\mathbf{x}'})$
end for
$I \leftarrow$ list of indices obtained from sorting $\{c_j\}_{1 \leq j \leq n}$ in ascending order (ties broken arbitrarily).
$S \leftarrow \emptyset$
for $1 \leq j \leq n$ do
    $S \leftarrow S \cup I[j]$
    if $\mathcal{N}(\mathbf{x}^{S|\mathbf{x}'}) \leq t$ then return $S$
end for
```

of the gradients in the sum of equation 4 with these values and show the statement of the lemma by instantiating equation 2 with $M = \sum_{j=1}^{d_{L-1}} M_j$, $A_m = W_j^L A_{m_j}^j$, $B_i = \pm\, B_i^j$ and $g_m(\mathbf{x}) = \pm\, g_{m_j}^j(\mathbf{x}) \cdot (D\sigma^L)(h^L(\mathbf{x}))$, where the sign in $\pm$ is determined by whether σ is monotonically increasing or decreasing. Again, by inductive hypothesis, it follows that $A_m \geq 0$ and $g_m(\mathbf{x}) \geq 0$ for each m and $\mathbf{x}$.

We now show equation 3. Let us consider the attribution for $\mathcal{N}$ defined in equation 1. Assume $i \neq j$ (otherwise the equation holds trivially). By replacing the value of the gradient in equation 1 with equation 2, we obtain $C_i^{\mathcal{N}}(\mathbf{x}, \mathbf{x}') = B_i(x_i - x_i') \sum_{m=1}^{M} A_m \int_0^1 g_m(\mathbf{x}' + \tau(\mathbf{x} - \mathbf{x}'))\mathrm{d}\tau$. Since integrated gradients satisfy the completeness and zero contribution axioms, we can compute the difference $\mathcal{N}(\mathbf{x}^{\{i\}|\mathbf{x}'}) - \mathcal{N}(\mathbf{x}^{\{j\}|\mathbf{x}'})$ as the sum of contributions $C_i^{\mathcal{N}}(\mathbf{x}^{\{i\}|\mathbf{x}'}, \mathbf{x}^{\{j\}|\mathbf{x}'})$ and $C_j^{\mathcal{N}}(\mathbf{x}^{\{i\}|\mathbf{x}'}, \mathbf{x}^{\{j\}|\mathbf{x}'})$ to obtain $(x_i' - x_i) \int_0^1 (\nabla\mathcal{N})_i \left(\mathbf{p}^{ij}(\tau)\right) \mathrm{d}\tau - (x_j' - x_j) \int_0^1 (\nabla\mathcal{N})_j \left(\mathbf{p}^{ij}(\tau)\right) \mathrm{d}\tau$. equation 2 provides the values for the gradients $(\nabla\mathcal{N})_i(\mathbf{p}^{ij}(\tau))$ and $(\nabla\mathcal{N})_j(\mathbf{p}^{ij}(\tau))$, which we can replace in the previous expression to finally derive equation 3. □

5.2 A Greedy Algorithm for Computing Minimal Feature Removal Sets

In this section, we propose a greedy algorithm for computing a minimal feature removal set for a given prediction of a monotonic FCN.

Algorithm 1 proceeds as detailed next. In each iteration of the first loop, the algorithm sets each individual input feature to the baseline value (while leaving the remaining components unchanged) and applies the input FCN to the resulting vector. The values obtained by each of these applications of the FCN are then sorted in ascending order. In the second loop, the algorithm successively assigns the components of $\mathbf{x}$ to the baseline in the order established in the previous step until the prediction no longer holds. The algorithm then returns S consisting of all features that were set to the baseline. Our algorithm is quadratic in the number of input features: both loops require linearly many applications of the FCN, and each application is feasible in linear time in the number of features Goodfellow et al. (2016). The algorithm's correctness relies on equation 3 in Lemma 1, which ensures that, when set to the baseline, each of the features selected by the algorithm in the second loop yields the largest change (amongst all other possible feature choices) in the evaluation of the network, thus bringing the network's output as close as possible to the prediction threshold. As a result, the output feature removal set S is guaranteed to contain a smallest number of features.

Theorem 2. *Algorithm 1 computes a minimal feature removal set for $\mathcal{N}(\mathbf{x}) > t$ relative to $\mathbf{x}'$.*

Proof. It suffices to show that, for each $j \in \{1, ..., n\}$, the choice of $I[j]$ in the second loop yields the largest change in the evaluation of $\mathcal{N}$. That is, for each $1 \leq j \leq n$ and $j \leq k \leq n$ we have $\mathcal{N}(\mathbf{x}^{S|\mathbf{x}'}) - \mathcal{N}(\mathbf{x}^{(S \cup I[j])|\mathbf{x}'}) \geq \mathcal{N}(\mathbf{x}^{S|\mathbf{x}'}) - \mathcal{N}(\mathbf{x}^{(S \cup I[k])|\mathbf{x}'})$. By construction of list I, the inequality $\mathcal{N}(\mathbf{x}^{I[j]|\mathbf{x}'}) - \mathcal{N}(\mathbf{x}^{I[k]|\mathbf{x}'}) \leq 0$ holds for each $1 \leq j \leq k \leq n$. We apply equation 3 in Lemma 1 together with the fact that $A_m \geq 0$ and $g_m(\mathbf{x}) \geq 0$ for each m and $\mathbf{x}$ (and hence $\int_0^1 g_m(\mathbf{p}^{ij}(\tau))\mathrm{d}\tau \geq 0$) to obtain $(B_{I[j]}(x'_{I[j]} - x_{I[j]}) - B_{I[k]}(x'_{I[k]} - x_{I[k]})) \leq 0$. Since $\{I[j], I[k]\} \subseteq \{1, ..., n\} - S$, we have $\left(\mathbf{x}^{S|\mathbf{x}'}\right)_{I[j]} = x_{I[j]}$ and $\left(\mathbf{x}^{S|\mathbf{x}'}\right)_{I[k]} = x_{I[k]}$. By applying equation 3 and the previous inequality, we finally obtain $\mathcal{N}(\mathbf{x}^{(S\cup I[j])|\mathbf{x}'}) \leq \mathcal{N}(\mathbf{x}^{(S\cup I[k])|\mathbf{x}'})$. □

Note that, although the correctness of our algorithm relies on the properties of integrated gradients, the algorithm itself only relies on the ability to apply the input network as a 'black box' and hence does not involve the computation of any attribution values.

6 Generalisation to Convolutional and Graph Neural Networks

In this section, we generalise our approach to more advanced neural architectures. Our main result is that, under monotonicity assumptions, Algorithm 1 continues to work verbatim if the input network $\mathcal{N}$ is a standard Convolutional (or Graph) Neural Network. In particular, we require that the weights are non-negative and that the non-linearities in $\mathcal{N}$ can be equivalently expressed as vector fields that are continuous, differentiable almost everywhere and with a Jacobian that is either at least 0 everywhere or at most 0 everywhere; the latter is the natural generalisation of the monotonicity requirement to vector fields, and it is satisfied by the standard non-linearities included in practical neural architectures.

To establish this result, we start by introducing as a theoretical tool the notion of a *generalised FCN*, where activation functions are defined as vector fields (and hence are no longer restricted to scalar functions applied component-wise to vectors). As discussed later on, this will give us the flexibility needed to mathematically capture the application of CNNs as well as certain variants of GNNs.

Definition 3. *A* generalised FCN *(GFCN) with $L \geq 1$ layers and input dimension $n \in \mathbb{N}$ is a tuple $\mathcal{N} = \langle \{\mathbf{W}^\ell\}_{1\leq\ell\leq L}, \{\mathbf{b}^\ell\}_{1\leq\ell\leq L}, \{\sigma^\ell\}_{1\leq\ell\leq L\}}\rangle$. For each $\ell \in \{1,\dots,L\}$, integers $d_\ell \in \mathbb{N}$ and $D_\ell \in \mathbb{N}$ represent the width and the pre-activation width of layer ℓ, respectively; we require $d_L = 1$ and define $d_0 = n$; matrix $\mathbf{W}^\ell \in \mathbb{R}^{D_\ell \times d_{\ell-1}}$ is a real-valued weight matrix; vector $\mathbf{b}^\ell \in \mathbb{R}^{D_\ell}$ is the bias vector; and the activation function $\sigma^\ell : \mathbb{R}^{D_\ell} \mapsto \mathbb{R}^{d_\ell}$ is a polytime-computable vector field.*

The application of a GFCN $\mathcal{N}$ to a vector $\mathbf{x} \in \mathbb{R}^n$ is defined as given in the preliminaries for FCNs.

The monotonicity requirement for FCNs can be seamlessly lifted to GFCNs, by establishing the natural requirements on the Jacobian of the each activation function in the network. In particular, we require the values of the Jacobian to be either non-negative everywhere or at most 0 everywhere.

Definition 4. *A GFCN $\mathcal{N} = \langle \{\mathbf{W}^\ell\}_{1\leq\ell\leq L}, \{\mathbf{b}^\ell\}_{1\leq\ell\leq L}, \{\sigma^\ell\}_{1\leq\ell\leq L\}}\rangle$ is* monotonic *if for each layer $1 \leq \ell \leq L$, the weights $\mathbf{W}^\ell$ are non-negative and the Jacobian $\mathbf{J}^\ell$ of $\sigma^\ell : \mathbb{R}^{D_\ell} \mapsto \mathbb{R}^{d_\ell}$ satisfies one of the following properties:*

- $(\mathbf{J}^\ell(\mathbf{y}))_{ij} \geq 0$ *for each $1 \leq i \leq d_\ell$, each $1 \leq j \leq D_\ell$, and each vector $\mathbf{y} \in \mathbb{R}^{D_\ell}$,* OR
- $(\mathbf{J}^\ell(\mathbf{y}))_{ij} \leq 0$ *for each $1 \leq i \leq d_\ell$, each $1 \leq j \leq D_\ell$, and each vector $\mathbf{y} \in \mathbb{R}^{D_\ell}$.*

We next show that Algorithm 1 remains correct if the input network is a monotonic GCFN. This is so because the generalised monotonicity requirement in Definition 4 ensures that equation 2 in Lemma 1 remains true if $\mathcal{N}$ is a monotonic GFCN.

Theorem 3. *Algorithm 1 computes a minimal feature removal set for $\mathcal{N}(\mathbf{x}) > t$ relative to $\mathbf{x}'$ when applied to a monotonic GFCN $\mathcal{N}$, vectors $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$ and threshold $t \in \mathbb{R}$ such that $\mathcal{N}(\mathbf{x}) > t$.*

Proof. It suffices to show that the statement in Lemma 1 also holds if $\mathcal{N}$ is a GFCN and, in particular, that equation 2 remains true. To this end, we set up a similar induction on the number L of layers in $\mathcal{N}$ as we did in the proof of Lemma 1. If $\mathcal{N}$ has a single layer ($L = 1$), then $\mathcal{N} = \langle \mathbf{W}, b, \sigma\rangle$, with $\mathbf{W}$ an $(D_1 \times n)$-matrix, b an D_1-dimensional vector and σ a scalar field $\sigma : \mathbb{R}^{D_1} \mapsto \mathbb{R}$. By the chain rule, each component of the gradient of $\mathcal{N}$ is given by $(\nabla\mathcal{N})_i(\mathbf{x}) = \sum_{j=1}^{D_1} W^1_{i,j}\left(\mathbf{J}^1\left(\mathbf{W}^1 \cdot \mathbf{x} + b^1\right)\right)_{1,j}$, where $\mathbf{J}^1$ is the Jacobian of σ^1 which, in this case, is a row vector. This expression is of the form equation 2 with $M = D_1$, $A_m = 1$, $B_i = \pm\, W^1_{i,j}$ and $g_m(\mathbf{x}) = \pm\ \left(\mathbf{J}^L\left(\sigma^L\left(\mathbf{W}^1 \cdot \mathbf{x} + b^1\right)\right)\right)_{1,j}$ where the sign $\pm$ is chosen depending on the type of monotonicity of the Jacobian (c.f. Definition 4). Observe that $g_m(\mathbf{x}) \geq 0$ by the monotonicity assumption from Definition 4.

For the inductive case, assume equation 2 holds for each GFCN with $L-1$ layers. The application of $\mathcal{N}$ with L layers to $\mathbf{x}$ is given by $\mathcal{N}(\mathbf{x}) = \sigma^L\left(\mathbf{h}^L(\mathbf{x})\right)$; thus, by the chain rule, each component of the gradient $(\nabla\mathcal{N})_i$ is now given as the sum $\sum_{j=1}^{D_L}\sum_{k=1}^{d_{L-1}} W_{k,j}^L(\nabla\mathcal{N}^k)_i(\mathbf{x})\left(\mathbf{J}^L\left(\mathbf{h}^L(\mathbf{x})\right)\right)_{1,j}$. Here, $\mathcal{N}^k = \{\mathbf{W}^\ell\}_{1\leq\ell\leq L-1}, \{\mathbf{b}^\ell\}_{1\leq\ell\leq L-1}, \{\sigma^\ell\}_{1\leq\ell\leq L-2}, \sigma_k^{L-1}\rangle$, where $\sigma_k^{L-1}: \mathbb{R}^{D_{L-1}} \mapsto \mathbb{R}$ is the k-th component of the vector field $\sigma^{L-1}: \mathbb{R}^{D_{L-1}} \mapsto \mathbb{R}^{d_{L-1}}$.

Following the proof of Lemma 1, the application of the inductive hypothesis to GFCN $\mathcal{N}^k$ yields $\left(\nabla\mathcal{N}^k\right)_i(\mathbf{x}) = B_i^k\ \sum_{m_k=1}^{M_k} A_{m_k}^k\ g_{m_k}^k(\mathbf{x})$. As a result, $(\nabla\mathcal{N})_i$ can be put into form equation 2 by taking $M = \sum_{j=1}^{D_L}\sum_{k=1}^{d_{L-1}} M_k$, $A_m = A_{m_k}^k W_k^L \geq 0$, $B_i = \pm\ B_i^k$, and $g_m(\mathbf{x}) = \pm\ g_{m_k}^k(\mathbf{x}) \cdot \left(J^L\left(h^L(\mathbf{x})\right)\right)_{1,j}$. The latter is again non-negative by assumption on the Jacobian. □

We finally argue that GFCNs are powerful enough to capture CNNs with non-negative weights (and certain variants of monotonic GNNs with non-negative weights) in a way that ensures correctness of Algorithm 1 when applied directly to such neural architectures, provided that their non-linearities satisfy our monotonicity requirements.

Consider a CNN $\mathcal{C}$ and a fixed dimension of input images. We argue that there is a GFCN $\mathcal{N}$ simulating the application of $\mathcal{C}$ to any image with the given dimensions. Thus, correctness of Algorithm 1 applied to $\mathcal{C}$ is implied by the correctness of the algorithm applied to $\mathcal{N}$ stated in Theorem 3. To see this, first note that the application of convolutional filters within a layer can be simulated by matrix multiplication Goodfellow et al. (2016). In turn, pooling operations can be seen as non-linearities returning a vector of smaller dimension, hence the definition of σ^ℓ as a vector field. Batch normalisation in inference mode can be seen as a linear transformation. Finally, pooling operations composed with standard non-linear activation functions can be simulated by first applying pooling, then multiplying by the identity matrix, and finally applying the standard activation function.

Similarly, GFCNs can also capture basic graph neural networks for which the feature update function is defined as $h_u^\ell = \sigma\left(\sum_{v\in\mathcal{N}(u)\cup\{u\}} \mathbf{W}^\ell h_v^{\ell-1}\right)$, where h_u^ℓ denotes the vector of features of node u at layer ℓ, and $\mathcal{N}(u)$ denotes the neighbours of node u in the input graph. We argue that, given such GNN $\mathcal{G}$ and a fixed topology of input graphs (i.e., a fixed set of nodes, edges, and fixed dimensions of the feature vector associated to each node), there exists a GFCN $\mathcal{N}$ simulating the application of $\mathcal{G}$ to any graph with the given topology. To see this, note that one can flatten the input so that each component of the input space corresponds to a feature of a particular node. The aggregation over neighbourhoods can be captured by matrix multiplication where the matrix has zeros where the corresponding nodes are not in the same neighbourhood, and the matrix has the corresponding weight values where the nodes are in the same neighbourhood. The non-linear activation function is then applied component-wise to each node feature, so it can be reproduced by applying the activation function component-wise to each component of the transformed input. This correspondence also generalises to Graph Convolutional Networks (GCNs) Kipf & Welling (2017). We note, however, that GFCNs cannot adequately capture arbitrary message-passing GNNs since there exists aggregatation functions which cannot be simulated by matrix multiplication followed by the application of a non-linear map.

We can conclude that our greedy algorithm can be directly applied to CNNs and GNNs with non-negative weights used in practice by using the networks as a 'black box' (and hence without the need of explicitly constructing the corresponding GFCN). In particular, the non-linearities used in practice (max-pooling, average-pooling, ReLu, sigmoid, tanh) satisfy our requirements.

7 Conclusion and Future Work

We have proposed to study a combinatorial problem, the *minimal feature removal problem*, inspired by ideas from attribution-based and perturbation-based explanation methods.

Our intractability result constitutes a theoretical limitation of what can be achieved by neural network explanations.

Our minimal feature removal sets can be efficiently computed for certain neural architectures, and could be used to probe the reliability of the predictions on certain tasks. Furthermore, minimal feature removal set sizes can be used as a measure of robustness of predictions. In line with existing work Alvarez Melis & Jaakkola (2018); Erion et al. (2021); Ismail et al. (2021), our approach could also be incorporated in the training process to improve network's robustness by generating additional training examples.

Scalability remains a potential issue in our approach since since our greedy algorithm is quadratic in the number of features; we expect, however, that parallelising predictions can help in that regard.

We hope that our work can motivate further theoretical analysis of neural network explainability.

REFERENCES

J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL `https://proceedings.neurips.cc/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf`.

D. Alvarez Melis and T. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

M. Ancona, E. Ceolini, C. Oztireli, and M. Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Sy21R9JAW`.

M. Ancona, C. Oztireli, and M. Gross. Explaining deep neural networks with a polynomial time algorithm for shapley values approximation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 72, 2019.

R. J. Aumann and L.S. Shapley. *Values of Non-Atomic Games*. Princeton University Press, 1974.

S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10, 2015.

M. Bajaj, L. Chu, Z. Y. Xue, J. Pei, L. Wang, P. Cho-Ho Lam, and Y. Zhang. Robust counterfactual explanations on graph neural networks. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

José Ramón Cano, Pedro Antonio Gutiérrez, Bartosz Krawczyk, Michal Wozniak, and Salvador García. Monotonic classification: An overview on algorithms, performance measures and data sets. *Neurocomputing*, 341:168–182, 2019. doi: 10.1016/j.neucom.2019.02.024. URL `https://doi.org/10.1016/j.neucom.2019.02.024`.

N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017. doi: 10.1109/SP.2017.49.

C.-H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud. Interpreting neural network classifications with variational dropout saliency maps. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

D. J. T. Cucala, Bernardo Cuenca Grau, Egor V. Kostylev, and Boris Motik. Explainable GNN-based models over knowledge graphs. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=CrCvGNHAIrz`.

P. Dabkowski and Y. Gal. Real time image saliency for black box classifiers. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive ex- planations with pertinent negatives. *Advances in Neural Information Processing Systems*, 32, 2018.

A.-K. Dombrowski, M. Alber, C. J. Anders, M. Ackermann, K.-R. Muller, and P. Kessel. Explanations can be manipulated and geometry is to blame. *Advances in Neural Information Processing Systems*, 33, 2019.

G. Erion, J.D. Janizek, P. Sturmfels, S. M. Lundberg, and S. I. Lee. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature Machine Intelligence*, 3:620–631, 2021.

R. C. Fong and A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

E. Friedman. Paths and consistency in additive cost sharing. *Internation Journal of Game Theory*, 32:501–518, 2004.

A. Ghorbani, A. Abid, and J. Zou. Interpretation of neural networks is fragile. *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings of 2005 IEEE International Joint Conference on Neural Networks*, volume 2, 2005. doi: 10.1109/IJCNN.2005.1555942.

Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee. Counterfactual visual explanations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 2376–2384, 2019.

A. Hannun, C. Case, J. Casper, G. Diamos B. Catanzaro, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HJz6tiCqYm`.

H. Hosseini, S. Kannan, and R. Poovendran. Dropping pixels for adversarial robustness. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 91–97, 2019. doi: 10.1109/CVPRW.2019.00017.

A. A. Ismail, H. Corrada Bravo, and S. Feizi. Improving deep learning interpretability by saliency guided training. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

G. Katz, C. Barrett, D.L. Dill, K. Julian, and M.J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification*, pp. 97–117. Springer International Publishing, 2017.

E. Kazim and A. S. Koshiyama. A high-level overview of ai ethics. *Patterns*, 2, 2021. doi: 10.1016/j.patter.2021.100314.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, 2017.

S. Kotyan and D.V. Vargas. Adversarial robustness assessment: Why in evaluation both l0 and l infinity attacks are necessary. *PLOS ONE*, 17(4):1–22, 2022. doi: 10.1371/journal.pone.0265723.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deepconvolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

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

J. Li, X. Liu, J. Zhao, and F. Shen. Autoadversary: A pixel pruning method for sparse adversarial attack. 2022. doi: 10.48550/ARXIV.2203.09756.

O. Li, H. Liu, C. Chen, and C. Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151, pp. 4499–4511, 2022.

João Marques-Silva, Thomas Gerspacher, Martin C. Cooper, Alexey Ignatiev, and Nina Narodytska. Explanations for monotonic classifiers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7469–7479. PMLR, 2021. URL `http://proceedings.mlr.press/v139/marques-silva21a.html`.

G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017. doi: https://doi.org/10.1016/j.patcog.2016.11.008.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

L. Shapley. *Contributions to the Theory of Games II*. Princeton University Press, 1953. doi: 10.1515/9781400881970-018.

A. Shrikumar, P. Greenside, and P. Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 3145–3153. PMLR, 2017.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, I. Antonoglou J. Schrittwieser, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529, 2016.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations*, 2014.

M. Sundararajan and A. Najmi. The many shapley values for model explanation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 9269–9278. PMLR, 2020. URL `https://proceedings.mlr.press/v119/sundararajan20b.html`.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 3319–3328. PMLR, 2017. URL `https://proceedings.mlr.press/v70/sundararajan17a.html`.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

C.-K. Yeh, C.-Y. Hsieh, A. Suggala, D. I. Inouye, and P. K. Ravikumar. On the (in)fidelity and sensitivity of explanations. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

X. Zhang, A. Solar-Lezama, and R. Singh. Interpreting neural network judgments via minimal, stable, and symbolic corrections. In *Advances in Neural Information Processing Systems*, volume 31, 2018.