The Minimal Feature Removal Problem in Neural Networks

Ouns El Harzli, Bernardo Cuenca Grau, Ian Horrocks
Department of Computer Science, University of Oxford
Oxford, United Kingdom
Correspondance to ouns.elharzli@new.ox.ac.uk

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 [31, 22, 32, 38, 41]. 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 [27].

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 [11, 13]; attribution-based methods assign a score to input features quantifying their contribution to the prediction relative to a baseline [44, 43, 4]; example-based methods explain predictions by retrieving training examples that are most similar to the given input [29, 34]; and perturbation-based methods generate corrections to an input causing the model to change its output [47, 21, 55, 7]. Evaluating and improving the robustness of neural predictions is also an important and challenging problem [6, 23]. 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 [45, 20]. 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 [18, 14, 46].
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 minimal 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 $\ell_0$-perturbation [30], where the goal is to change the prediction by identifying changes to a discrete set of input features; indeed, as discussed in [33, 24], $\ell_0$-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 [47, 35, 7, 16, 21] 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 [44], 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) [19, 31] and graph neural networks (GNNs) [20, 28] 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 [36, 8, 11].

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 $S \subseteq \{1, \ldots, n\}$ of their components, we denote with $\mathbf{x}|_S$ 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 $f : \mathbb{R}^n \to \mathbb{R}$, 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 $\mathcal{N} = \{(\mathbf{W}^\ell)_1 \leq \ell \leq L, (\mathbf{b}^\ell)_1 \leq \ell \leq L, (\sigma^\ell)_1 \leq \ell \leq L\}$. For each layer $\ell \in \{1, \ldots, L\}$, the integer $d_\ell \in \mathbb{N}$ is the width of layer $\ell$ and we require $d_0 = 1$ 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 $\sigma^\ell : \mathbb{R} \to \mathbb{R}$ is a polytime-computable *activation function* applied component-wise to vectors.
The application of network $\mathcal{N}$ to an input feature vector $x \in \mathbb{R}^n$ generates a sequence $x^1, \ldots, x^L$ of vectors defined as $x^\ell = \sigma^\ell(h^\ell)$, where $x^0 = x$ and $h^\ell = W^\ell \cdot x^{\ell-1} + b^\ell$. The result $\mathcal{N}(x)$ of applying $\mathcal{N}$ to $x$ is the scalar $x^L$. Thus, the neural network realises a function $\mathcal{N} : \mathbb{R}^n \mapsto \mathbb{R}$.

When for each $\ell \in \{1, \ldots, L\}$, $\sigma^\ell$ 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 $x$ is given by the result of comparing $\mathcal{N}(x)$ to the threshold $t$.

### 3 Background on Attribution-based Methods

Attribution methods \cite{23, 24, 39} are a family of explanation techniques which, given as input function $f : \mathbb{R}^n \mapsto \mathbb{R}$, a vector $x \in \mathbb{R}^n$ and a baseline vector $x' \in \mathbb{R}^n$, assign a numerical score or contribution $C^f_i(x, x')$ to each component $i \in \{1, \ldots, n\}$. Attribution methods are often designed to fulfill some (or all) of the following axioms for all functions $\mathbb{R}^n \mapsto \mathbb{R}$ and vectors $x, x' \in \mathbb{R}^n$, components $1 \leq i \leq n$ and coefficients $\lambda_1, \lambda_2 \in \mathbb{R}$.

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

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 \cite{39} is one of the most popular thanks to its nice properties. Calculating Shapley values is, however, intractable, which has motivated research on approximation techniques \cite{24}. Other popular attribution methods have been designed specifically for neural networks; these include Layer-wise Relevance Propagation \cite{6}, DeepLIFT \cite{40}, Deep Taylor decompositions \cite{37}, and Saliency Maps \cite{42, 1, 12, 10}.

We will exploit the properties of Integrated Gradients \cite{44, 5}, 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 $x$ and baseline vector $x'$ is defined as follows:

$$C^f_i(x, x') := (x_i - x'_i) \int_0^1 (\nabla f)_{i}(x' + \tau(x - x')) \, d\tau.$$ (1)

Integrated gradients is the only path-based attribution method satisfying all of the aforementioned axioms \cite{17}. 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 $x$, we aim at identifying a smallest set of ‘most relevant’ features for the prediction. Similarly to attribution-based methods, we consider an uninformative baseline $x'$ 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 $x$, 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 $N$ be a FCN with $L$ layers and input dimension $n$, let $t$ be a numeric threshold, let $x \in \mathbb{R}^n$ be such that $N(x) > t$, and let $x' \in \mathbb{R}^n$ be a baseline vector. A feature removal set for $N(x) > t$ relative to $x'$ is a subset $S \subseteq \{1, \ldots, n\}$ satisfying $N(x^S|x') \leq t$. The size $|S|$ of $S$ is the number of elements it contains. Furthermore, we say $S$ is minimal if no $S' \subseteq \{1, \ldots, n\}$ such that $|S'| < |S|$ is a feature removal set for $N(x) > t$ relative to $x'$.

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 [44].

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 $x, x' \in \mathbb{R}^n$, a ReLU FCN $N$ of input dimension $n$, a numeric threshold $t$ such that $N(x) > t$, and $1 \leq k \leq n$, decide whether there exists a feature removal set for $N(x) > t$ relative to $x'$ 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 $a_1, \ldots, a_m$ and $\xi$, 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 $0_m$ the $m$-dimensional null vector and with $1_m$ the $m$-dimensional column vector with all components equal to 1.

We map an instance $a_1, \ldots, a_m, \xi$ of SUBSET-SUM to an instance of our problem by setting $k = m$, $t = 0$, $x = 1_{m+1}$, $x' = 0_{m+1}$ and $N$ the 2-layer ReLU network defined as next. In the first layer, $W^1$ 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 $1_{m+1}$ in the third row; furthermore, $h^1_i = -\sum_{j=1}^{m} a_j - \xi$, $b^1_i = \sum_{j=1}^{m} a_j + \xi$ and $b^1_m = -m$. In the second layer, $W^2 = 1_3$, $b^2 = 0$. With input $x = 1_{m+1}$, we have that $h^2_1 = -\xi$, $h^2_2 = \xi$, and $h^2_3 = 1$ which yields that $h^2 \geq 1$. By construction, we thus have $N(x) > 0$, 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 $N(x) > 0$ relative to $x'$ of size at most $k = m$. By definition, $N(x^S|x') \leq 0$. We claim that, by construction of $N$, $S$ 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 $h^1_1 = -h^2_1$, this yields $h^1_1 = h^1_2 = 0$. By construction, with input $x^S$, we have $h^1_1 = \sum_{i \in S} a_i - \xi = -(\xi - \sum_{i \in S} a_i)$. It follows that $S$ is a solution to the $\sum_{i \in S} a_i = \xi$. 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 $x^S$, 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 $N(x^S|x') \leq 0$.

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 $N(x^S|x') \leq t$, 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.
We show equation \(2\) by induction on the number of elements in the sum. In turn, as we show next, the monotonicity requirement ensures that each component of the gradient of \(N\) at \(x\) can be expressed as a sum where (1) the number of components in the sum is fixed for \(N\) (i.e., it does not depend on \(x\)) 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 \(x\) but which is always non-negative, and two coefficients that do not depend on \(x\).

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 \(x\) to the baseline, we are not altering the relative order of the integrated gradient contributions (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 \(N\) with input dimension \(n\), there exists a non-negative integer \(M\), positive coefficients \(\{A_m\}_{1 \leq m \leq M}\), real coefficients \(\{B_i\}_{1 \leq i \leq n}\), and functions \(\{g_m\}_{1 \leq m \leq M}\) from \(\mathbb{R}^n\) to \(\mathbb{R}_{\geq 0}\), such that the following identities are satisfied for each \(x, x' \in \mathbb{R}^n\) and each \(i, j \in \{1, \ldots, n\}\):

\[
(\nabla N)_i(x) = B_i \sum_{m=1}^{M} A_m g_m(x), \quad \text{and} \quad (2)
\]

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

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

**Proof.** We show equation \(2\) by induction on the number of layers in \(N\). If \(L = 1\), then \(N = (W, b, \sigma)\) with \(W\) an \(n\)-dimensional vector, \(b\) a scalar and \(\sigma: \mathbb{R} \mapsto \mathbb{R}\). Using the chain rule, we have \((\nabla N)_i(x) = W_i \cdot (D\sigma)(W \cdot x + b)\), where \(D\sigma\) is the derivative of \(\sigma\) in Euler’s notation. This expression is of the form equation \(2\) with \(M = 1\), \(A_1 = 1 \geq 0\), \(B_i = W_i\) if \(\sigma\) is monotonically increasing (dually, \(B_i = -W_i\) if \(\sigma\) is monotonically decreasing), and \(g_1(x) = (D\sigma)(W \cdot x + b)\) if \(\sigma\) is monotonically increasing (dually, \(g_1(x) = -(D\sigma)(W \cdot x + b)\) if \(\sigma\) is monotonically decreasing).

In both cases, monotonicity of \(\sigma\) ensures that \(g_1(x) \geq 0\) for any \(x\).

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

\[
(\nabla N)_i(x) = (\nabla h^L)_i(x) \cdot (D\sigma^L)(h^L(x)) = \sum_{j=1}^{d_{L-1}} W_j^L \cdot (\nabla N^j)_i(x) \cdot (D\sigma^L)(h^L(x)). \quad (4)
\]

Here, \(N^j\) is specified by \(\{W^\ell\}_{1 \leq \ell \leq L-2}, W_j^{L-1}, \{b^\ell\}_{1 \leq \ell \leq L-2}, b_j^{L-1}, \{\sigma^\ell\}_{1 \leq \ell \leq L-1}\), where \(W_j^{L-1}\) and \(b_j^{L-1}\) are the \(j\)-th row of \(W^{L-1}\) and the \(j\)-th element of \(b^{L-1}\), respectively.

We apply the inductive hypothesis to obtain the value of the gradient for each \(1 \leq j \leq d_{L-1}\), which is given by \((\nabla N^j)_i(x) = B_i^j \sum_{m_j=1}^{M_{j}} A_{m_j}^j g_{m_j}^j(x)\). But now, we can replace the value of the gradients in the sum of equation \(4\) with these values and show the statement of the lemma.
Algorithm 1

**Input:** monotonic FCN $N$, vectors $x, x' \in \mathbb{R}^n$ and threshold $t \in \mathbb{R}$ such that $N(x) > t$.

for $1 \leq j \leq n$ do
  $c_j \leftarrow N(x^{(j)}|x')$
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
  $S \leftarrow S \cup \{j\}$
  if $N(x^{S|\bar{x}'}) \leq t$ then return $S$
end for

by instantiating equation 2 with $M = \sum_{j=1}^{L-1} M_j$, $A_m = W^L_j A_{m_j}$, $B_i = \pm B^i_j$ and $g_m(x) = \pm g_{m_j}(x) \cdot (D \varphi^L)(h^L(x))$, where the sign in $\pm$ is determined by whether $\sigma$ is monotonically increasing or decreasing. Again, by inductive hypothesis, it follows that $A_m \geq 0$ and $g_m(x) \geq 0$ for each $m$ and $x$.

We now show equation 3. Let us consider the attribution for $N$ defined in equation 1. Assume $i \neq j$ (otherwise the attribution holds trivially). By replacing the value of the gradient in equation 1 with equation 2 we obtain $C_i^N(x, x') = B_i(x_i - x'_i) \sum_{m=1}^M A_m \int_0^1 g_m(x + \tau(x - x')) d\tau$.

Since integrated gradients satisfy the completeness and zero contribution axioms, we can compute the difference $N(x^{(i)}|x') - N(x^{(j)}|x')$ as the sum of contributions $C_i^N(x^{(i)}|x', x^{(j)}|x')$ and $C_j^N(x^{(i)}|x', x^{(j)}|x')$ to obtain $(x'_i - x_i) \int_0^1 (\nabla N)_i (p^j(\tau)) d\tau - (x'_j - x_j) \int_0^1 (\nabla N)_j (p^j(\tau)) d\tau$.

Theorem 2. Algorithm 1 computes a minimal feature removal set for $N(x) > t$ relative to $x'$.

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 $N$. That is, for each $1 \leq j \leq n$ and $1 \leq k \leq n$ we have $N(x^{S|\bar{x}'}) - N(x^{S \cup \{j\}|x'}) \geq N(x^{S|\bar{x}'}) - N(x^{S \cup \{k\}|x'})$. By construction of list $I$, the inequality $N(x^{I[j]|x'}) - N(x^{I[k]|x'}) \leq 0$ 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 $g_m(x) \geq 0$ for each $m$ and $x$ (and hence $\int_0^1 g_m(p^j(\tau)) d\tau \geq 0$) 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 $(x^{S|\bar{x}'})_{I[j]} = x_{I[j]}$ and $(x^{S|\bar{x}'})_{I[k]} = x_{I[k]}$. By applying equation 3 and the previous inequality, we finally obtain $N(x^{(S \cup \{j\})|x'}) \leq N(x^{(S \cup \{j\})|x'})$. 

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 $x$ 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 $|X|$. 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 (among 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.
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 $\mathcal{N} = \{\{\mathbf{W}^l\}_{1 \leq l \leq L}, \{\mathbf{b}^l\}_{1 \leq l \leq L}, \{\sigma^l\}_{1 \leq l \leq L}\}$. For each $l \in \{1, \ldots, L\}$, integers $d_l \in \mathbb{N}$ and $D_l \in \mathbb{N}$ represent the width and the pre-activation width of layer $l$, respectively; we require $d_l = 1$ and define $d_0 = n$; matrix $\mathbf{W}^l \in \mathbb{R}^{D_l \times d_{l-1}}$ is a real-valued weight matrix; vector $\mathbf{b}^l \in \mathbb{R}^{D_l}$ is the bias vector; and the activation function $\sigma^l : \mathbb{R}^{D_l} \rightarrow \mathbb{R}^{d_l}$ 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 $\ell$ everywhere; the latter is the natural generalisation of the monotonicity requirement to vector fields.

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

- $(J^l(y))_{ij} \geq 0$ for each $1 \leq i \leq d_l$, each $1 \leq j \leq D_l$, and each vector $y \in \mathbb{R}^{D_l}$, OR
- $(J^l(y))_{ij} \leq 0$ for each $1 \leq i \leq d_l$, each $1 \leq j \leq D_l$, and each vector $y \in \mathbb{R}^{D_l}$.

We next show that Algorithm 1 remains correct if the input network is a monotonic GFCN. 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 $\mathcal{N}(x) > t$ relative to $x'$ when applied to a monotonic GFCN $\mathcal{N}$, vectors $x, x' \in \mathbb{R}^n$ and threshold $t \in \mathbb{R}$ such that $\mathcal{N}(x) > t$.

**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 $\mathcal{N} = (\mathbf{W}, b, \sigma)$, with $\mathbf{W}$ an $(D_1 \times n)$-matrix, $b$ an $D_1$-dimensional vector and $\sigma$ a scalar field $\mathbb{R}^{D_1} \rightarrow \mathbb{R}$. By the chain rule, each component of the gradient of $\mathcal{N}$ is given by $(\nabla \mathcal{N})_i(x) = \sum_{j=1}^{D_1} W_{i,j} J^1(\mathbf{W}^1 \cdot x + b^1)_j$, where $J^1$ is the Jacobian of $\sigma^1$ which, in this case, is a row vector. This expression is of the form equation 2 with $M = D_1$, $A_m = 1$, $B_i = \pm W_{i,j}$ and $g_m(x) = \pm (J^1(\sigma^1(\mathbf{W}^1 \cdot x + b^1)))_{i,j}$ where the sign $\pm$ is chosen depending on the type of monotonicity of the Jacobian (c.f. Definition 4). Observe that $g_m(x) \geq 0$ 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 $x$ is given by $\mathcal{N}(x) = \sigma^L(h^L(x))$; thus, by the chain rule, each component of the gradient $(\nabla \mathcal{N})_i$ is now given as the sum $\sum_{j=1}^{D_L} \sum_{k=1}^{d_{L-1}} W_{k,j}^L (\nabla \mathcal{N}^k)_i(x) (J^L(h^L(x)))_{1,j}$. 
Here, $\mathcal{N}^k = \{\mathbf{W}^\ell\}_{1 \leq \ell \leq L-1}, \{\mathbf{b}^\ell\}_{1 \leq \ell \leq L-1}, \{\sigma^\ell\}_{1 \leq \ell \leq L-2}, \sigma^L_{k-1}$, where $\sigma^L_{k-1} : \mathbb{R}^{D_{L-1}} \mapsto \mathbb{R}$ is the $k$-th component of the vector field $\sigma^L_{k-1} : \mathbb{R}^{D_{L-1}} \mapsto \mathbb{R}^{d_{L-1}}$.

Following the proof of Lemma[1] the application of the inductive hypothesis to GFCN $\mathcal{N}^k$ yields $(\nabla \mathcal{N})^k_1(x) = B^k_{i \ell} \sum_{m_k=1}^{M_k} A^k_{m_k} \cdot g^k_{m_k}(x)$. As a result, $(\nabla \mathcal{N})^k_1$ can be put into form equation 2 by taking $M = \sum_{j=1}^{D_1} \sum_{k=1}^{d_{L-1}} M_k, A_m = A^k_{m_k}W^L_k \geq 0, B_i = \pm B^k_i, \text{and } g_m(x) = \pm g^k_{m_k}(x) \cdot (J^L(h^L(x)))_{1, j}$. The latter is again non-negative by assumption on the Jacobian. \hfill \qedsymbol

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 [19]. In turn, pooling operations can be seen as non-linearities returning a vector of smaller dimension, hence the definition of $\sigma^\ell$ 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 $h_u^\ell$ denotes the vector of features of node $u$ at layer $\ell$, 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) [28]. We note, however, that GFCNs cannot adequately capture arbitrary message-passing GNNs since there exists aggregation 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 [2, 15, 25], 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 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

[1] J. Adebayo et al. “Sanity Checks for Saliency Maps”. In: Advances in Neural Information Processing Systems. Vol. 31. 2018. URL: https://proceedings.neurips.cc/paper/2018/file/29a8ed24b1ad22ec2e7e049b8737-Paper.pdf.

[2] D. Alvarez Melis and T. Jaakkola. “Towards Robust Interpretability with Self-Explaining Neural Networks”. In: Advances in Neural Information Processing Systems. Vol. 31. 2018.

[3] 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. Vol. 72. 2019.

[4] M. Ancona et al. “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.

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

[6] S. Bach et al. “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: PLoS ONE 10 (7 2015).

[7] M. Bajaj et al. “Robust Counterfactual Explanations on Graph Neural Networks”. In: Advances in Neural Information Processing Systems. Vol. 34. 2021.

[8] José Ramón Cano et al. “Monotonic classification: An overview on algorithms, performance measures and data sets”. In: Neurocomputing 341 (2019), pp. 168–182. DOI: 10.1016/j.neucom.2019.02.024 URL: https://doi.org/10.1016/j.neucom.2019.02.024.

[9] N. Carlini and D. Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: 2017 IEEE Symposium on Security and Privacy (SP). 2017, pp. 39–57. DOI: 10.1109/SP.2017.49.

[10] C.-H. Chang et al. “Interpreting Neural Network Classifications with Variational Dropout Saliency Maps”. In: Advances in Neural Information Processing Systems. Vol. 30. 2017.

[11] D. J. T. Cucala et al. “Explainable GNN-Based Models over Knowledge Graphs”. In: International Conference on Learning Representations. 2022. URL: https://openreview.net/forum?id=CrCvGNHAIrz.

[12] P. Dabkowski and Y. Gal. “Real Time Image Saliency for Black Box Classifiers”. In: Advances in Neural Information Processing Systems. Vol. 30. 2017.

[13] A. Dhurandhar et al. “Explanations based on the missing: Towards contrastive explanations with pertinent negatives”. In: Advances in Neural Information Processing Systems 32 (2018).

[14] A.-K. Dombrowski et al. “Explanations can be manipulated and geometry is to blame”. In: Advances in Neural Information Processing Systems 33 (2019).

[15] G. Erion et al. “Improving performance of deep learning models with axiomatic attribution priors and expected gradients”. In: Nature Machine Intelligence 3 (2021), 620–631.

[16] 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.

[17] E. Friedman. “Paths and consistency in additive cost sharing”. In: International Journal of Game Theory 32 (2004), pp. 501–518.

[18] A. Ghorbani, A. Abid, and J. Zou. “Interpretation of Neural Networks Is Fragile”. In: Thirty-Third AAAI Conference on Artificial Intelligence (2019).

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

[20] 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. Vol. 2. 2005. DOI: 10.1109/IJCNN.2005.1555942.

[21] G. Erion et al. “Improving performance of deep learning models with axiomatic attribution priors and expected gradients”. In: Nature Machine Intelligence 3 (2021), 620–631.

[22] 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.

[23] E. Friedman. “Paths and consistency in additive cost sharing”. In: International Journal of Game Theory 32 (2004), pp. 501–518.

[24] A. Ghorbani, A. Abid, and J. Zou. “Interpretation of Neural Networks Is Fragile”. In: Thirty-Third AAAI Conference on Artificial Intelligence (2019).

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

[26] 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. Vol. 2. 2005. DOI: 10.1109/IJCNN.2005.1555942.
[21] Y. Goyal et al. “Counterfactual Visual Explanations”. In: Proceedings of the 36th International Conference on Machine Learning. Vol. 97. 2019, pp. 2376–2384.

[22] A. Hannun et al. “Deep speech: Scaling up end-to-end speech recognition.” In: arXiv preprint arXiv:1412.5567 (2014).

[23] 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

[24] D. Hendrycks and T. Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations”. In: International Conference on Learning Representations. 2019.

[25] 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). 2019, pp. 91–97. DOI: 10.1109/CVPRW.2019.00017

[26] A. Ismail, H. Corrada Bravo, and S. Feizi. “Improving Deep Learning Interpretability by Saliency Guided Training”. In: Advances in Neural Information Processing Systems. Vol. 34. 2021.

[27] G. Katz et al. “Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks”. In: Computer Aided Verification. Springer International Publishing, 2017, pp. 97–117.

[28] E. Kazim and A. S. Koshiyama. “A high-level overview of AI ethics”. In: Patterns 2 (9 2021). DOI: 10.1016/j.patter.2021.100314

[29] T. N. Kipf and M. Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: International Conference on Learning Representations (2017).

[30] P. W. Koh and P. Liang. “Understanding black-box predictions via influence functions”. In: Proceedings of the 34th International Conference on Machine Learning. Vol. 70. 2017.

[31] S. Kotyan and D.V. Vargas. “Adversarial robustness assessment: Why in evaluation both L0 and L infinity attacks are necessary”. In: PLOS ONE 17.4 (2022), pp. 1–22. DOI: 10.1371/journal.pone.0265723

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks.” In: Advances in Neural Information Processing Systems 25 (2012).

[33] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning”. In: Nature 521.7553 (2015), pp. 436–444.

[34] J. Li et al. “AutoAdversary: A Pixel Pruning Method for Sparse Adversarial Attack”. In: (2022). DOI: 10.48550/ARXIV.2203.09756

[35] O. Li et al. “Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions”. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018).

[36] A. Lucic et al. “CF-GNNExplainer: Counterfactual Explanations for Graph Neural Networks”. In: Proceedings of The 25th International Conference on Artificial Intelligence and Statistics. Vol. 151. 2022, pp. 4499–4511.

[37] Jo˜ ao Marques-Silva et al. “Explanations for Monotonic Classifiers”. In: Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 7469–7479. URL: http://proceedings.mlr.press/v139/marques-silva21a.html

[38] G. Montavon et al. “Explaining nonlinear classification decisions with deep Taylor decomposition”. In: Pattern Recognition 65 (2017), pp. 211–222. DOI: https://doi.org/10.1016/j.patcog.2016.11.008

[39] J. Schmidhuber. “Deep learning in neural networks: An overview”. In: Neural networks 61 (2015), pp. 85–117.

[40] A. Shrikumar, P. Greenside, and P. Kundaje. “Learning Important Features Through Propagating Activation Differences”. In: Proceedings of the 34th International Conference on Machine Learning. Vol. 70. PMLR, 2017, pp. 3145–3153.

[41] D. Silver et al. “Mastering the game of Go with deep neural networks and tree search.” In: Nature 529 (2016).

[42] 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.
[43] M. Sundararajan and A. Najmi. “The Many Shapley Values for Model Explanation”. In: Proceedings of the 37th International Conference on Machine Learning. Vol. 119. PMLR, 2020, pp. 9269–9278. URL: https://proceedings.mlr.press/v119/sundararajan20b.html

[44] M. Sundararajan, A. Taly, and Q. Yan. “Axiomatic Attribution for Deep Networks”. In: Proceedings of the 34th International Conference on Machine Learning. Vol. 70. PMLR, 2017, pp. 3319–3328. URL: https://proceedings.mlr.press/v70/sundararajan17a.html

[45] C. Szegedy et al. “Intriguing properties of neural networks”. In: International Conference on Learning Representations. 2014.

[46] C.-K. Yeh et al. “On the (In)fidelity and Sensitivity of Explanations”. In: Advances in Neural Information Processing Systems. Vol. 32. 2019.

[47] 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. Vol. 31. 2018.
This figure "all_images_mnist_7_2.png" is available in "png" format from:

http://arxiv.org/ps/2205.09901v2
This figure "all_images_mnist_7v2_2.png" is available in "png" format from:

http://arxiv.org/ps/2205.09901v2
This figure "all_images_mnist_8_2.png" is available in "png" format from:

http://arxiv.org/ps/2205.09901v2
This figure "all_images_mnist_8_3.png" is available in "png" format from:

http://arxiv.org/ps/2205.09901v2
This figure "all_images_mnist_3_5.png" is available in "png" format from:

http://arxiv.org/ps/2205.09901v2
This figure "all_images_mnist_6_5.png" is available in "png" format from:

http://arxiv.org/ps/2205.09901v2
This figure "all_images_mnist2_7.png" is available in "png" format from:

http://arxiv.org/ps/2205.09901v2
This figure "all_images_mnist_6_9.png" is available in "png" format from:

http://arxiv.org/ps/2205.09901v2