Pathwise Explanation of ReLU Neural Networks

Seongwoo Lim[1] **Won Jo**[2] **Joohyung Lee**[3,4] **Jaesik Choi**[2,5]

[1]Ulsan National Institute of Science and Technology (UNIST)
[2]Korea Advanced Institute of Science and Technology (KAIST)
[3]Arizona State University, [4]Samsung Research, [5]INEEJI

Abstract

Neural networks have demonstrated a wide range of successes, but their "black box" nature raises concerns about transparency and reliability. Previous research on ReLU networks has sought to unwrap these networks into linear models based on activation states of all hidden units. In this paper, we introduce a novel approach that considers subsets of the hidden units involved in the decision making path. This pathwise explanation provides a clearer and more consistent understanding of the relationship between the input and the decision-making process. Our method also offers flexibility in adjusting the range of explanations within the input, i.e., from an overall attribution input to particular components within the input. Furthermore, it allows for the decomposition of explanations for a given input for more detailed explanations. Experiments demonstrate that our method outperforms others both quantitatively and qualitatively.

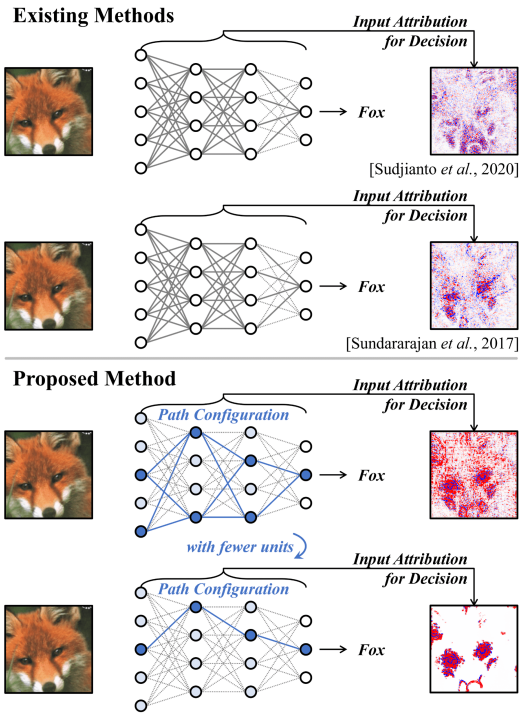


Figure 1: Comparison of the methods

1 Introduction

Neural networks have demonstrated a wide range of successes in various domains (Caruana *et al.*, 2015; Litjens *et al.*, 2017; Yurtsever *et al.*, 2020; Zhu *et al.*, 2016). However, many of these networks are perceived as "black boxes" due to the opacity of their decision-making processes. This has led to the rise of eXplainable Artificial Intelligence (XAI), which seeks to clarify how these black box models operate (Adadi and Berrada, 2018; Arrieta *et al.*, 2020; Das and Rad, 2020). XAI is vital in ensuring the reliability of practical applications (Gunning and Aha, 2019; Gunning *et al.*, 2019), diagnosing malfunctions in neural networks (Lapuschkin *et al.*, 2019), and promoting consistent operation in our daily lives.

ReLU is widely employed as an activation function due to its advantages, including mitigating the vanishing gradient problem and enabling efficient computation (Goodfellow *et al.*, 2016). It is known that a Feed-Forward Neural Network with ReLU can be represented as a piecewise linear model (Sattelberg *et al.*, 2020). Sudjianto *et al.* (2020) introduce a process called *unwrapping*, which leverages this property to describe a ReLU NN as a collection of local linear functions based on *activation patterns*—that is, the activation states of all hidden units within the network. Consequently, a ReLU NN is represented by a distinct linear model for inputs that correspond to the same activation pattern.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

Code: https://github.com/Jo-won/PathwiseExplanation

In this paper, we also represent ReLU NN as a piecewise linear model. However, instead of focusing on activation patterns, we consider activation states for individual *paths*—specific subsets of hidden units involved in a particular decision process. This approach facilitates a more explicit explanation of the relationship between the input and the corresponding *paths* they traverse for a decision, in contrast to prior works.[1]

Additionally, the concept of a *path* offers flexibility in adjusting the range of explanations within the input. Specifically, by controlling the maximum number of units included in the path configuration process for each layer, as shown in Figure 1, our method is capable of not only providing an overall attribution of the input to the prediction but also facilitating the explanation of particular components within the input. We reveal that this adjustment can provide more detailed explanations for the decision by enabling the decomposition of explanations for a given input.

Furthermore, we demonstrate that our pathwise explanation offers better consistency in relation to the input compared to existing methodologies. This enhanced consistency arises from the utilization of only a subset of hidden units rather than the entire set. In our experiments on the use of linear models for recognizing informative attributions, our linear model derived from various types of paths outperforms others both quantitatively and qualitatively.

Our contributions are as follows:

- We introduce a *pathwise* explanation for ReLU NNs along with its associated algorithms for computing paths. The pathwise explanation describes a clearer relationship between the input and the corresponding decision-making by the use of *paths* (Section 3).
- The pathwise explanation facilitates the decomposition of explanations for a given input (e.g. *Fox*) into its individual components (e.g., *Fox*'s eyes and ears). Furthermore, it elucidates the underlying reasons for incorrect predictions made by the model (Section 5.1 and 5.2).
- In experiments recognizing informative attributions for explaining specific inputs, we demonstrate that our method with various paths outperforms others both quantitatively and qualitatively (Section 5.3).

2 Related Work

As mentioned, our work is related to Sudjianto *et al.* (2020). However, in contrast to their approach that considers all activated hidden units, we focus on specific subsets that form paths leading to the decision process. (Villani and McBurney, 2023) extends *unwrapping* from Sudjianto *et al.* (2020) to NNs with diverse structures, such as Graph Neural networks and tensor convolutional networks.

[1] See Section 4 for more details.

Input attribution methods, designed to compute an input's contribution to the output—either through a heatmap or as a linear model (Ribeiro *et al.*, 2016)—are widely utilized for interpreting NNs. Among them, gradient-based input attribution techniques offer intuitive means to elucidate the roles of hidden units.

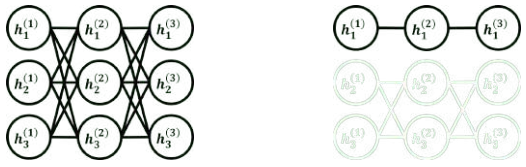
The Saliency method employs the gradient of the input for a specific target unit (Simonyan *et al.*, 2013). This gradient inherently signifies the input's contributions to the target unit. Moreover, it becomes possible to modify the input to enhance target units using this gradient, as described in (Mordvintsev *et al.*, 2015). The goal is often to identify input sections that amplify the target unit's response, as opposed to suppressing it. Techniques such as Guided Backpropagation (Springenberg *et al.*, 2014) and Deconvnet (Zeiler and Fergus, 2014) leverage the ReLU activation function to counteract the effects of negative weights. The Integrated Gradient method (Sundararajan *et al.*, 2017), adhering to both sensitivity and implementation invariance standards, computes the integral of the gradient from a baseline to the input. Class Activation Mapping (CAM) (Zhou *et al.*, 2016) increases the model's transparency by using Global Average Pooling (GAP) rather than fully-connected. GradCAM (Selvaraju *et al.*, 2017) is an extension of CAM that can generate explanations for differentiable models without using GAP. Another class of input attribution methods revolves around perturbation-based approaches. For instance, Occlusion method (Zeiler and Fergus, 2014) modifies a specific rectangular input region to a baseline and observes consequent output changes.

Some studies produce model explanations leveraging the unique properties of the ReLU function. Notably, in a neural network's Taylor series expansion with ReLU activation, explanations can be derived by solely considering the first-order term; this is because higher-order terms become zero with ReLU (Bach *et al.*, 2015; Montavon *et al.*, 2017).

3 Pathwise Explanation of ReLU NN

In this section, we introduce the concept of a *path* within a NN with ReLU activation. First, we establish the idea of a *one-way complete path*, representing a straightforward linear model case. Subsequently, we expand the definition to accomodate multi-way paths.

We consider a Feed-Forward NN with a ReLU activation function in each layer, which is termed as a ReLU NN. For an input $X \in \mathbb{R}^{d_0}$, where d_0 represents the vectorized input dimension, a ReLU NN $f : \mathbb{R}^{d_0} \to \mathbb{R}^{d_{N+1}}$ computes the prediction $f(X)$ with the ouput dimension d_{N+1}. For the i^{th} layer, $layer_i$, a vector of hidden units $h^{(i)} = [h_1^{(i)}, ..., h_{d_i}^{(i)}]$ is given before ReLU activation, where d_i is the number of hidden units in $layer_i$. A weight matrix between $layer_{i-1}$ and $layer_i$ is denoted as $W^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$,



(a) Original network (b) One-way complete path

(c) Multi-way incomplete path (d) Not a path

Figure 2: Examples of paths with three hidden layers

and a weight associating the j^{th} hidden unit in $layer_{(i-1)}$ and the k^{th} hidden unit in $layer_i$ is denoted as $W^{(i)}_{k,j} \in \mathbb{R}$. The term $*$ is used as an index for all hidden units. For example, $W^{(i)}_{*,j} \in \mathbb{R}^{d_i \times 1}$ represents weights connecting the j^{th} hidden unit in $layer_{i-1}$ to all hidden units in $layer_i$. A bias vector for $layer_i$ is denoted as $b^{(i)} \in \mathbb{R}^{d_i}$.

A ReLU NN can be represented as an undirected graph, with hidden units serving as nodes and weights as edges. In this context, we say that a set of hidden units is *connected* in a ReLU NN if the subgraph they induce is connected.

Definition 1. A ***path*** is defined as a set of hidden units *connected* in a ReLU NN.

- A ***one-way path*** is defined as a set of hidden units with no more than one unit per layer (in contrast to a ***multi-way path***).
- A ***complete path*** is defined as a set of hidden units with at least one unit in every layer (in contrast to an ***incomplete path***).
- A ***path is activated*** when all units in the path have positive values with regard to the input; otherwise, the ***path is deactivated***.
- The ***depth of a path*** is the number of layers that include hidden units in the path.

Example 1. For the ReLU NN with 3 hidden layers in Figure 2 (a), $\{h^{(1)}_1, h^{(2)}_1, h^{(3)}_1\}$ represents a *one-way complete path* with a depth of 3, and $\{h^{(1)}_1, h^{(1)}_2, h^{(2)}_1\}$ is a *multi-way incomplete path* with a depth of 2. Note that $\{h^{(1)}_1, h^{(3)}_1\}$ is not a *path* because $h^{(1)}_1$ and $h^{(3)}_1$ are not connected.

3.1 Formalization of *One-way Complete Path*

The ReLU function can be expressed as

$$ReLU(x) = x \cdot \phi(x),$$

where $\phi(x) = 1$ for $x > 0$, $\phi(x) = 0$ for $x \leq 0$. ϕ can be applied elementwise for a multi-dimensional input. A Feed-Forward ReLU NN can be represented as follows:

$$\begin{aligned} h^{(1)} &= W^{(1)}X + b^{(1)} \\ h^{(2)} &= W^{(2)}(h^{(1)} \cdot \phi(h^{(1)})) + b^{(2)} \\ &\dots \\ h^{(N)} &= W^{(N)}(h^{(N-1)} \cdot \phi(h^{(N-1)})) + b^{(N)} \\ f(X) &= W^{(N+1)}(h^{(N)} \cdot \phi(h^{(N)})) + b^{(N+1)}. \end{aligned} \tag{1}$$

For each hidden unit, Equation (1) can be decomposed:

$$\begin{aligned} h^{(1)}_{i_1} &= W^{(1)}_{i_1,*}X + b^{(1)}_{i_1} \\ h^{(n)}_{i_n} &= \sum_{k=1}^{d_{n-1}} W^{(n)}_{i_n,k} h^{(n-1)}_k \phi(h^{(n-1)}_k) + b^{(n)}_{i_n} \\ f(X) &= \sum_{k=1}^{d_N} W^{(N+1)}_{*,k} h^{(N)}_k \phi(h^{(N)}_k) + b^{(N+1)}, \end{aligned}$$

where $0<i_1 \in \mathbb{N} \leq d_1$, $0<i_n \in \mathbb{N} \leq d_n$ and $1<n \in \mathbb{N}<N-1$. Furthermore, $f(X)$ can be *unfolded* as

$$\begin{aligned} f(X) = &\sum_{i_1,i_2,..,i_N} W^{(N+1)}_{*,i_N} W^{(N)}_{i_N,i_{N-1}} \dots W^{(2)}_{i_2,i_1} W^{(1)}_{i_1,*} X \prod_{j=1}^{N} \phi(h^{(j)}_{i_j}) \\ &+ \sum_{i_1,i_2,..,i_N} W^{(N+1)}_{*,i_N} W^{(N)}_{i_N,i_{N-1}} \dots W^{(2)}_{i_2,i_1} b^{(1)}_{i_1} \prod_{j=1}^{N} \phi(h^{(j)}_{i_j}) \\ &+ \sum_{i_2,..,i_N} W^{(N+1)}_{*,i_N} W^{(N)}_{i_N,i_{N-1}} \dots W^{(3)}_{i_3,i_2} b^{(2)}_{i_2} \prod_{j=2}^{N} \phi(h^{(j)}_{i_j}) \\ &\dots \\ &+ \sum_{i_N} W^{(N+1)}_{*,i_N} b^{(N)}_{i_N} \prod_{j=N}^{N} \phi(h^{(j)}_{i_j}) + b^{(N+1)}, \end{aligned} \tag{2}$$

where $0<i_n \in \mathbb{N} \leq d_n$.

One of the primary concepts in this paper is representing a ReLU NN as a piecewise linear model based on a path derived from Equation (2). The ReLU function, due to its piecewise linear nature, allows for the expression that is a sum of piecewise linear models comprised of ϕ for each hidden unit. Given a one-way complete path $\mathbf{p} = [h^{(1)}_{i_1}, ..., h^{(N)}_{i_N}]$, we define the terms $W^{\mathbf{p}}$ and $b^{\mathbf{p}}$ as follows:

$$\begin{aligned} W^{\mathbf{p}} &= W^{(N+1)}_{*,i_N} W^{(N)}_{i_N,i_{(N-1)}} \dots W^{(2)}_{i_2,i_1} W^{(1)}_{i_1,*} \\ b^{\mathbf{p}} &= W^{(N+1)}_{*,i_N} W^{(N)}_{i_N,i_{(N-1)}} \dots W^{(2)}_{i_2,i_1} b^{(1)}_{i_1}, \end{aligned} \tag{3}$$

where $1 \leq i_j \in \mathbb{N} \leq d_j$. Note that in Equation (2), these $W^{\mathbf{p}}$ and $b^{\mathbf{p}}$ are multiplied by $\prod_{h \in \mathbf{p}} \phi(h)$—the product of ϕ for all hidden units in $\mathbf{p}$. As a result, we define the *piecewise linear model $f^{\mathbf{p}}(X)$ for the path* $\mathbf{p}$ as

$$f^{\mathbf{p}}(X) = (W^{\mathbf{p}}X + b^{\mathbf{p}}) \prod_{h \in \mathbf{p}} \phi(h). \tag{4}$$

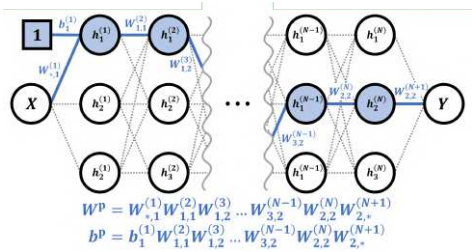


Figure 3: An illustration of Proposition 1

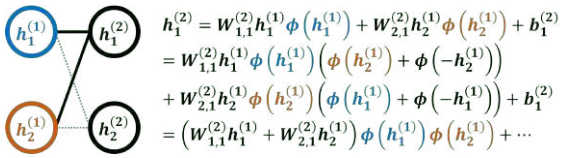


Figure 4: An illustration of Proposition 2

Proposition 1. *For a one-way complete path* $\mathbf{p}$*, the piecewise linear model* $f^{\mathbf{p}}(X)$ *constitutes a term in Equation* (2) *and is non-zero if and only if the path is activated (the proof is in Appendix A.1).*

3.2 Generalization of Linear Model for Multi-way Complete Path

The preceding section discussed a one-way complete path. Given the piecewise linear nature of a ReLU NN, the extension to multi-way complete paths is straightforward. As previously, we define the piecewise linear model for a multi-way complete path $\mathbf{p}$ in the following form:

$$f^{\mathbf{p}}(X) = (W^{\mathbf{p}}X + b^{\mathbf{p}}) \prod_{h \in \mathbf{p}} \phi(h) \tag{5}$$

where $W^{\mathbf{p}} \in \mathbb{R}^{d_{N+1} \times d_0}$, $b^{\mathbf{p}} \in \mathbb{R}^{d_{N+1}}$.

For a multi-way complete path $\mathbf{p}$ for a ReLU NN, we define $W^{\mathbf{p}}$ and $b^{\mathbf{p}}$ for equation (4) by summing up the weights and biases of all possible one-way complete paths that are subsets of the multi-way complete path $\mathbf{p}$.

$$W^{\mathbf{p}} = \sum_{\substack{p \text{ is a one-way complete path} \\ \text{that is a subset of } \mathbf{p}}} W^p \tag{6}$$

$$b^{\mathbf{p}} = \sum_{\substack{p \text{ is a one-way complete path} \\ \text{that is a subset of } \mathbf{p}}} b^p \tag{7}$$

Proposition 2. *For a multi-way complete path* $\mathbf{p}$*, the piecewise linear model* $f^{\mathbf{p}}(X)$ *represents a summation of terms from Equation* (2)*, and is non-zero if and only if the path is activated.*

The proof of Proposition 2 (Appendix A.2) uses the following property of a ReLU function:

$$\phi(x) + \phi(-x) = 1 \tag{8}$$

Algorithm 1 Multi-way Path Construction

Input: h: set of hidden units in N layer ReLU NN,
  $h^{(N+1)}$: NN output, $targetIndex$: target index,
  $depth$: depth of the generated path
  $width$: width of the generated path
  α: threshold for importance
Initialize: $path^{(N+1)} = [1, 2, \ldots, |h^{(N+1)}|]$
  $W^{prev}{=}I$ ▷ identity matrix
for $n \leftarrow N$ to $(N{-}depth{+}1)$ **do**
  $path^{(n)} = emptyList()$
  $W = \frac{d}{dh^{(n)}} h^{(n+1)}$ ▷ $\mathbb{R}^{d_{n+1} \times d_n}$
  $W = W^{prev}(W[path^{(n+1)}, *])$ ▷ $\mathbb{R}^{d_{N+1} \times d_n}$
  $imp = matMul(W, diag(h^{(n)}))^T$ ▷ $\mathbb{R}^{d_n \times d_{N+1}}$
  $imp = softmax(imp)[*, targetIndex]$ ▷ $\mathbb{R}^{d_n}$
  $setOfTopKIndex{=}topKIndex(imp, K{=}width)$
  for $index$ in $setOfTopKIndex$ **do**
    if $imp[index] > \alpha$ **then**
      $path^{(n)}.add(h^{(n)}_{index})$
    end if
  end for
  $W^{prev} = W[*, path^{(n)}]$ ▷ $\mathbb{R}^{d_{N+1} \times |path^{(n)}|}$
end for
return $\cup_{n=(N-depth+1)}^{N} path^{(n)}$

which means that the hidden unit can have two cases: activated ($\phi(x){=}1$) or deactivated ($\phi(-x){=}1$). Figure 4 illustrates Proposition 2.

Example 2. As in Figure 4, consider the multi-way complete path $\mathbf{p} = [h_1^{(1)}, h_2^{(1)}, h_1^{(2)}, h_2^{(2)}]$.

$$\begin{aligned} W^{\mathbf{p}} &= \left(\frac{df(x)}{dh_1^{(2)}} \frac{dh_1^{(2)}}{dh_1^{(1)}} + \frac{df(x)}{dh_2^{(2)}} \frac{dh_2^{(2)}}{dh_1^{(1)}} \right) \frac{dh_1^{(1)}}{dx} \\ &\quad + \left(\frac{df(x)}{dh_1^{(2)}} \frac{dh_1^{(2)}}{dh_2^{(1)}} + \frac{df(x)}{dh_2^{(2)}} \frac{dh_2^{(2)}}{dh_2^{(1)}} \right) \frac{dh_2^{(1)}}{dx} \\ &= W^{[h_1^{(1)}, h_1^{(2)}]} + W^{[h_1^{(1)}, h_2^{(2)}]} + W^{[h_2^{(1)}, h_1^{(2)}]} + W^{[h_2^{(1)}, h_2^{(2)}]}. \end{aligned}$$

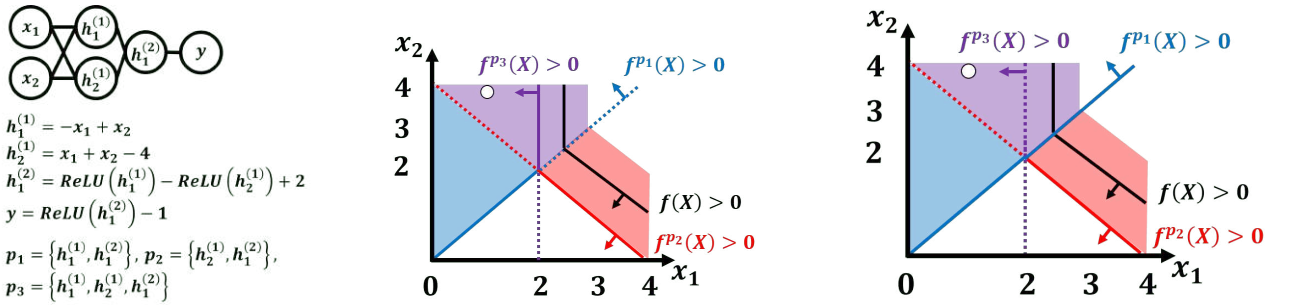
Example 3. Consider the ReLU NN, as in Figure 5. There exist three complete paths, $\mathbf{p}_1{=}\{h_1^{(1)}, h_1^{(2)}\}$, $\mathbf{p}_2{=}\{h_2^{(1)}, h_1^{(2)}\}$ and $\mathbf{p}_3{=}\{h_1^{(1)}, h_2^{(1)}, h_1^{(2)}\}$. Then,

$$\begin{aligned} f^{\mathbf{p}_1}(x_1, x_2) &= (-x_1 + x_2)\phi(h_1^{(1)})\phi(h_1^{(2)}), \\ f^{\mathbf{p}_2}(x_1, x_2) &= (-x_1 - x_2 + 4)\phi(h_2^{(1)})\phi(h_1^{(2)}), \\ f^{\mathbf{p}_3}(x_1, x_2) &= (-2x_1 + 4)\phi(h_1^{(1)})\phi(h_2^{(1)})\phi(h_1^{(2)}). \end{aligned}$$

The ReLU NN can be expressed as

$$\begin{aligned} f(x_1, x_2) &= f^{\mathbf{p}_1}(x_1, x_2) + f^{\mathbf{p}_2}(x_1, x_2) + 2\phi(h_1^{(2)}) - 1 \\ &= \begin{cases} f^{\mathbf{p}_1}(x_1, x_2) + 1, & \text{in the blue region} \\ f^{\mathbf{p}_2}(x_1, x_2) + 1, & \text{in the red region} \\ f^{\mathbf{p}_3}(x_1, x_2) + 1, & \text{in the purple region.} \end{cases} \end{aligned}$$

The explanation by $f^{\mathbf{p}_1}(x_1, x_2)$ means that x_1 has a negative contribution towards the classification of the *posi-*



(a) ReLU NN structure

(b) Inconsistent explanation for the white sample as *positive* class (y>0) by using $f^{\mathbf{p}_3}$=$f^{\mathbf{p}_1}$+$f^{\mathbf{p}_2}$

(c) Consistent explanation for the white sample as *positive* class by using $f^{\mathbf{p}_1}$ only.

Figure 5: Comparison of the proposed pathwise explanation and *unwrapping* through a pedagogical example. For the white sample, $f^{\mathbf{p}_1}$ and $f^{\mathbf{p}_2}$ predict different classes. A pathwise explanation can generate a more consistent explanation. In (b) and (c), a solid line represents a decision boundary relevant to that example, while a dotted line is included for reference purposes but does not directly relate to the example.

tive class (y>0), whereas x_2 has a positive contribution. Note that $f^{\mathbf{p}_3}(x_1, x_2)$=$f^{\mathbf{p}_1}(x_1, x_2)$ + $f^{\mathbf{p}_2}(x_1, x_2)$ within the purple region. Our proposed method enables explanations from three distinct paths: p_1, p_2, and p_3, individually for inputs within this purple region.

3.3 Path construction

This section presents an algorithm to find paths that are positively related to the target class prediction. Algorithm 1 is a pseudocode for our path construction method. We compute the weight between the $(n{+}1)^{th}$ layer and the n^{th} layer by gradient (line 10). The importance of the hidden units in the n^{th} layer is computed by $softmax(W^{(N+1)}h^{(N)})$ (lines 12–13). The top-k important units are those that increases the target class prediction value over predictions for other classes (lines 14–19). The $depth$, $width$, α are hyperparameters used to generate various paths. $width$ indicates the maximum number of hidden units used at each layer when constructing a path. α controls how much the prediction value for the target class is increased by the hidden units compared to other classes. We consistently set α at $\frac{1}{|classes|}$, where $|classes|$ represents the total number of classes in the dataset. This approach is adopted to identify neurons that contribute significantly more to the target class than the average contribution across all classes, thereby highlighting neurons with a substantial impact on the decision-making process.

The path generated by Algorithm 1 consists of hidden units from the $(N{-}depth{+}1)^{th}$ layer to the N^{th} layer. Then we may consider that the ReLU NN is partitioned into two subnetworks: f_1 that spans from the first layer to the $(N{-}depth{+}1)^{th}$ layer, and f_2 that spans from the $(N{-}depth{+}1)^{th}$ layer to the N^{th} layer. The path that is found by Algorithm 1 is incomplete for the original NN, but is complete for f_2. Thus, we can compute the linear model for this path. For the first subnetwork, we can employ direct linearization:

$$W_1 = \frac{d}{dX} f_1(X),$$
$$b_1 = f_1(X) - W_1 X.$$

Here, $f_1(X)$ denotes the first subnetwork, which is ReLU NN itself. We then obtain the linear model of the whole ReLU NN for the incomplete path $\mathbf{p}$ as a composite function of these linear models, i.e., $f_2^{\mathbf{p}}(W_1 X + b_1)$.

Example 4. Let's revisit the simple ReLU NN in Figure 5. We will show how the paths are constructed by the algorithm 1 (with $depth$=2 and $width$=1) for the white sample (x_1, x_2)=$(1.0, 4.0)$ in Figure 5 (b) as the *positive* class (y>0). For the white sample,

$$h_1^{(1)} = 3, \quad h_2^{(1)} = 1, \quad h_1^{(2)} = 4, \quad y = 3$$

For the second layer, W=$\begin{bmatrix}1\end{bmatrix}$. Then, the importance of $h_1^{(2)}$:

$$imp_{h_1^{(2)}} = \frac{e^{1\times h_1^{(2)}}}{e^{1\times h_1^{(2)}} + e^{-(1\times h_1^{(2)})}} = 0.982$$

We add the $h_1^{(2)}$ to $path^{(2)}$. Next, for the first layer,

$$W = \begin{bmatrix}1\end{bmatrix}\begin{bmatrix}1 & -1\end{bmatrix} = \begin{bmatrix}1 & -1\end{bmatrix}.$$

Then, we compute the importance of $h_1^{(1)}$ and $h_2^{(1)}$:

$$imp_{h_1^{(1)}} = \frac{e^{1\times h_1^{(1)}}}{e^{1\times h_1^{(1)}} + e^{-(1\times h_1^{(1)})}} = 0.953,$$

$$imp_{h_2^{(1)}} = \frac{e^{-1\times h_2^{(1)}}}{e^{-1\times h_2^{(1)}} + e^{-(-1\times h_2^{(1)})}} = 0.269.$$

We add $h_1^{(1)}$ to $path^{(1)}$. The constructed path for the white sample is $path^{(1)}\cup path^{(2)}$=$\{h_1^{(1)}, h_1^{(2)}\}$ (Figure 5 (c)).

4 Comparison with Unwrapping by Sudjianto *et al.*

Both our pathwise explanation and *unwrapping* (Sudjianto *et al.*, 2020; Villani and McBurney, 2023) use the activa-

tion states in a ReLU NN to represent the ReLU NN as a piecewise linear model. The fundamental distinction is that the pathwise explanation leverages the activation states of a subset of hidden units connected by paths, whereas *unwrapping* encompasses the activation states of all hidden units. The local linear model produced by *unwrapping* for an input aligns with the linear model of the path that includes all activated hidden units for that input along with an additional bias.

Example 5. Consider again the simple ReLU network in Figure 5 (a). According to Sudjianto *et al.* (2020), the input space is divided by activation patterns. For example, any input instance within the purple region in Figure 5 (b) activates $h_1^{(1)}$ & $h_2^{(1)}$ & $h_1^{(2)}$. Figure 5 (b) show a local linear model on each activation region. The weights of the linear model are used to generate explanations for the input's contribution to the prediction. However, this method often creates inconsistent explanations. For instance, the white input in Figure 5 (c) is classified as the *positive* class ($y > 0$) by $f^{\mathbf{p}_1}(x_1, x_2) + 1$. However, the same input is classified as *negative* class ($y < 0$) by $f^{\mathbf{p}_2}(x_1, x_2) + 1$. Consequently, the explanation by $f^{\mathbf{p}_1}(x_1, x_2)$ describes how the ReLU NN perceives the input as the *positive* class, while the explanation by $f^{\mathbf{p}_2}(x_1, x_2)$ describes the opposite. Thus, the explanation generated by $f^{\mathbf{p}_3}(x_1, x_2)$, which is the summation of $f^{\mathbf{p}_1}(x_1, x_2)$ and $f^{\mathbf{p}_2}(x_1, x_2)$, conveys mixed messages, representing both the *positive* and *negative* classification. However, as explained in Example 4, our method offers a consistent explanation for predicting the white input as the *positive* class by using $f^{\mathbf{p}_1}(x_1, x_2)$ instead of $f^{\mathbf{p}_3}(x_1, x_2)$ even for the purple region (Figure 5 (c)).

The Figure 1 shows the difference in explanations generated by *unwrapping* and pathwise explanation. From this figure, we can see that our method better captures the features of the input, such as the eyes and ears of the fox.

5 Experiments

This section evaluates the effectiveness of our pathwise explanation method. First, we demonstrate that multiple paths, or subsets of hidden units, can account for explanations of individual components within a given input by decomposing the model's decision-making process. In addition, it specifies the reasons for the incorrect responses generated by the model. Subsequently, we assess both the quantitative and qualitative explanations for the input, as calculated using our method.

We conducted experiments on a curated subset of the ImageNet dataset (Russakovsky *et al.*, 2015), which includes 10 distinct classes. For the purpose of explaining decisions, regardless of the method employed, all experiments consistently utilized the VGG-16 architecture (Simonyan and Zisserman, 2015). This architecture was fine-tuned using pretrained weights from ImageNet, with adjustments made to its classifier to match the number of classes in the subset.

Figure 6: An example of explanation decomposition through the paths. (left) *Fox* class input; (middle) an explanation by the path using the *Fox*'s eyes for the *Fox* class prediction; and (right) using the *Fox*'s ear.

Figure 7: An example of the explanation for the incorrect prediction through the paths. (left) *mantis* class input; (middle) input region supporting the ReLU prediction as *mantis*; (right) input region supporting the ReLU prediction as *prison*.

Notably, our method is applicable to CNNs, as both convolution and pooling operations are linear for a given input. Specifically, we can extend our method designed for MLPs to CNNs by equating a combined convolutional and pooling operation, $f : \mathbb{R}^{H_i \times W_i \times C_i}$, to a linear layer in an MLP, $g : \mathbb{R}^{H_i W_i C_i} \to \mathbb{R}^{H_o W_o C_o}$, which takes a flattened input and produces a flattened output. This approach has been utilized in all our experiments with CNNs, e.g., VGG-16 as detailed in Section 5 and ResNet-18 in Section F.

We compare our method with the following methods.

- **Saliency** (Simonyan *et al.*, 2013),
- Input×Gradient (**IxG**) (Shrikumar *et al.*, 2016),
- Integrated Gradients (**IGs**) (Sundararajan *et al.*, 2017),
- Guided Backprop. (**GBP**) (Springenberg *et al.*, 2014),
- Guided GradCAM (**GGC**) (Selvaraju *et al.*, 2017),
- Blur Integrated Gradients (**BlurIG**) (Xu *et al.*, 2020).

Note that the attributions provided by Saliency and IxG are exactly the same with the weight (W) and WX of the piecewise linear model obtained by *unwrapping*, respectively. The attribution map, generated by the pathwise explanation of path $\mathbf{p}$ for input X, is constructed as $W^{\mathbf{p}}X$.

5.1 Explanation Decomposition by Paths

A significant advantage of our pathwise explanation is its ability to identify multiple distinct paths that play a crucial role in the model's decision-making. Moreover, we observed that individual paths can effectively capture salient features within the input. Assuming that neurons in the

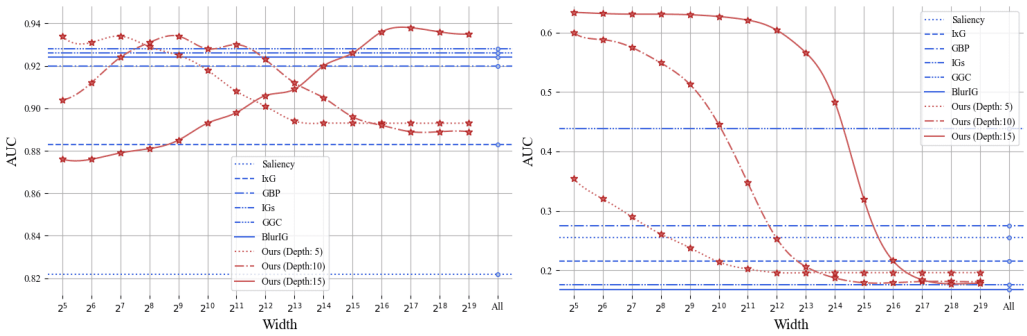


Figure 8: Quantitative explanation via *insertion* and *deletion* depending on the maximum number of units per layer (*width*) and the number of layers (*depth*) for path configuration. In our method, a *depth* of 15, which is the maximum, represents a complete path, while any lesser value indicates an incomplete path.

higher layers have high-level features like the fox's eye, we generated multiple paths that include each hidden neuron extracted in Algorithm 1 from the N^{th} layer. For example, if a $path^{(N)}=\{h_1, h_2\}$ is generated at step $n=N$ in Algorithm 1, we assumed $path^{(N)}=\{h_1\}$ and proceeded to the next step to create a path. We also assumed $path^{(N)}=\{h_2\}$ and proceeded to the next step to create another path.
In Figure 6, we present an illustrative example that showcases the model making an accurate prediction. In this instance, our proposed method identifies two distinct paths that adeptly capture essential features, such as the *Fox*'s eyes and ear, vital for correctly recognizing the input as a *Fox*. Moreover, our pathwise explanation facilitates the breakdown of each object within the input, allowing for the identification and explanation of multiple objects (Appendix D).

5.2 Explanation for Incorrect Prediction

Our path construction algorithm (Algorithm 1) is capable of generating explanations for any class. Particularly for incorrect predictions in ReLU NNs, these explanations offer valuable insights into the underlying reasons for the model's misclassifications. This is illustrated in Figure 7. In this figure, the middle image represents our explanation of the correct class (*mantis*), which the model predicted with the second-highest confidence. In contrast, the right image represents our explanation for the incorrect class (*prison*), which the model predicted with the highest confidence. These explanations suggest that the model's inaccurate predictions arise from similarities between the background of the input image and prison bars. Based on these observations, we conclude that the proposed explanation method can also be used to pinpoint the factors leading to the model's misclassifications.

5.3 Pixel Insertion and Deletion Game

To compare our attribution map with other explanation methods, we employed two causal metrics: *insertion* and *deletion* (Petsiuk *et al.*, 2018). In this context, the attribute map offers a quantified explanation of the impact that each input region (or each pixel, in the case of images) has on the decision. For our method, this influence is computed using the linear model for the path.

	Insertion(↑)	*Deletion*(↓)
Saliency	0.822	0.255
IxG	0.883	0.215
IGs	0.926	0.176
GBP	0.920	0.275
GGC	0.928	0.438
BlurIG	0.924	**0.168**
Ours	**0.936**	0.179

Table 1: Area Under Curve (AUC) of *Insertion* and *Deletion*. ↑ indicates that the larger the value, the better the explanation, whereas ↓ indicates the opposite.

These two causal metrics determine that the attribution map provides a better explanation when there is a more significant degree of change according to addition or removal as described in the map. In particular, *insertion* perceives a higher value as indicative of a better explanation, where the value is the Area Under Curve (AUC) for performance changes caused by increasing the proportion of inserting important pixels. Conversely, *deletion* interprets a lower value as a more proper explanation, obtained when increasing the proportion of removal.

As evidenced in Table 1, our method outperforms the rest in the *insertion* metric and offers competitive performance in the *deletion* metric. Such results underscore that our approach, distinct from others, excels at explaining all components within the input. Moreover, it facilitates elucidating each individual component, as described before.

5.4 Explanation by Various Types of Path

Various paths can be constructed by varying *depth* and *width* using Algorithm 1. Figures 8 and 9 demonstrate that the pathwise explanations for these various paths outperform other methods both quantitatively and qualitatively.

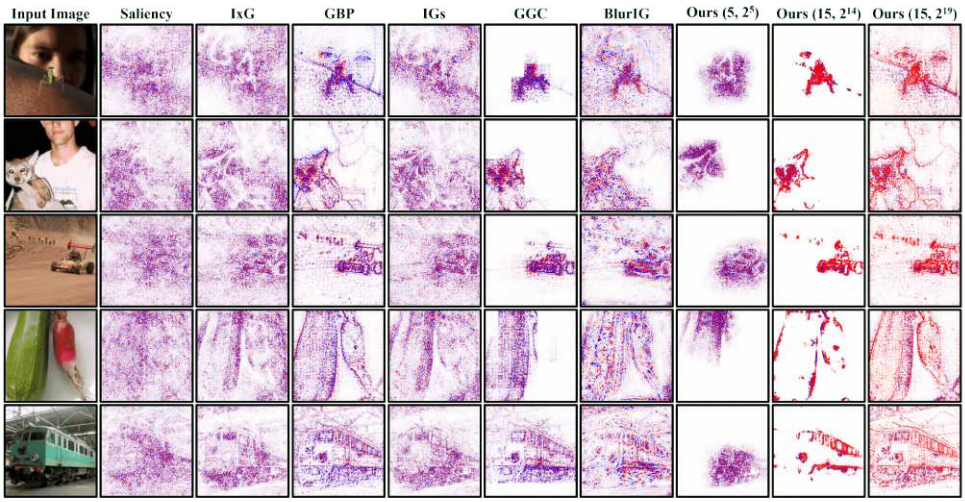


Figure 9: Qualitative explanation depending on the maximum number of units per layer (*width*) and the number of layers (*depth*) for path configuration. The expression enclosed in parentheses in the proposed method is as follows: (*depth*, *width*). Positive and negative attributions are indicated in red and blue. vspace-2mm

More precisely, in case of a small *depth*, since the path includes only high-level layers containing hidden units with relatively broad receptive fields, the path with a small *width* can sufficiently consider the entire area within an image. However, the quality of attribution produced by the path is degraded when the *width* increases, since leads to the inclusion of less significant units.

Conversely, in case of a large *depth*, a large *width* can enhance the upper bound increment of explanation by considering the entire area within an image. Figure 9 graphically presents our attribution maps as influenced by both *width* and *depth*. Interestingly, even when both *depth* and *width* are minimal, the pathwise explanation aptly pinpoints the object's location. For paths with the maximum depth, our attribution maps not only stand out more than those of other methods employing all units but also adeptly highlight key features as the *width* is adjusted.

6 Conclusion

We proposed a method of explaining a ReLU network in terms of piecewise linear model that corresponds to the *path*—the subset of hidden units. We demonstrated that the proposed method can generate various explanations for a single input by employing different paths. Additionally, we introduced a path construction algorithm that generates consistent explanations for the model's output. The experiment indicates that the pathwise explanation provides a clearer and more consistent understanding of the relationship between the input and the decision-making process over the others.

Acknowledgements

This work was partially supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00984, Development of Artificial Intelligence Technology for Personalized Plug-and-Play Explanation and Verification of Explanation; No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)), the National Science Foundation under Grant IIS-2006747, and Samsung Research.

References

Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6:52138–52160, 2018.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 2020.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015.

Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (XAI): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

David Gunning and David Aha. DARPA's explainable artificial intelligence (XAI) program. *AI magazine*, 2019.

David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. XAI—Explainable artificial intelligence. *Science robotics*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 2019.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 2017.

Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern recognition*, 65:211–222, 2017.

A. Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.

Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: randomized input sampling for explanation of black-box models. In *British Machine Vision Conference 2018, BMVC 2018*, page 151. BMVA Press, 2018.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 2015.

Ben Sattelberg, Renzo Cavalieri, Michael Kirby, Chris Peterson, and Ross Beveridge. Locally linear attributes of ReLU neural networks. *arXiv preprint arXiv:2012.01940*, 2020.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Agus Sudjianto, William Knauth, Rahul Singh, Zebin Yang, and Aijun Zhang. Unwrapping the black box of deep relu networks: Interpretability, diagnostics, and simplification. *arXiv preprint arXiv:2011.04041*, 2020.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, 2017.

Mattia Jacopo Villani and Peter McBurney. Unwrapping all relu networks. *arXiv preprint arXiv:2305.09424*, 2023.

Shawn Xu, Subhashini Venugopalan, and Mukund Sundararajan. Attribution in scale and space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9680–9689, 2020.

Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 2020.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, 2014.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classifica-

tion in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [No]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes] See the appendix
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [No]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] See the appendix

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] See the appendix
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Proofs

A.1 Proof of Proposition 1

Given a one-way complete path $\mathbf{p} = [h_{i_1}^{(1)}, ..., h_{i_N}^{(N)}]$, we define the terms $W^{\mathbf{p}}$ and $b^{\mathbf{p}}$ as follows:

$$W^{\mathbf{p}} = W_{*,i_N}^{(N+1)} W_{i_N,i_{(N-1)}}^{(N)} \dots W_{i_2,i_1}^{(2)} W_{i_1,*}^{(1)}$$
$$b^{\mathbf{p}} = W_{*,i_N}^{(N+1)} W_{i_N,i_{(N-1)}}^{(N)} \dots W_{i_2,i_1}^{(2)} b_{i_1}^{(1)}, \tag{i}$$

where $1 \leq i_j \in \mathbb{N} \leq d_j$. Thus, we define the *piecewise linear model* $f^{\mathbf{p}}(X)$ *for the path* $\mathbf{p}$ as

$$f^{\mathbf{p}}(X) = (W^{\mathbf{p}} X + b^{\mathbf{p}}) \prod_{h \in \mathbf{p}} \phi(h). \tag{ii}$$

Proposition 1. *For a one-way complete path* $\mathbf{p}$*, the piecewise linear model* $f^{\mathbf{p}}(X)$ *constitutes a term in Equation* (2) *and is non-zero if and only if the path is activated and* $(W^{\mathbf{p}} X + b^{\mathbf{p}})$ *is non-zero.*

Proof. From Equation (2), we take the terms including $\prod_{h \in \mathbf{p}} \phi(h)$ where $\mathbf{p} = [h_{i_1}^{(1)}, ..., h_{i_N}^{(N)}]$.

$$\left(W_{*,i_N}^{(N+1)} W_{i_N,i_{N-1}}^{(N)} \dots W_{i_2,i_1}^{(2)} W_{i_1,*}^{(1)} X + W_{*,i_N}^{(N+1)} W_{i_N,i_{N-1}}^{(N)} \dots W_{i_2,i_1}^{(2)} b_{i_1}^{(1)} \right) \prod_{j=1}^{N} \phi(h_{i_j}^{(j)})$$
$$= (W^{\mathbf{p}} X + b^{\mathbf{p}}) \prod_{h \in \mathbf{p}} \phi(h).$$

By Definition 1, "the path is activated" means that all hidden units in the path are activated given input X. Therefore,

$$f^{\mathbf{p}}(X) = (W^{\mathbf{p}} X + b^{\mathbf{p}}) \prod_{h \in \mathbf{p}} \phi(h) \text{ is non-zero}$$
$$\Rightarrow \prod_{h \in \mathbf{p}} \phi(h) = 1$$
$$\Rightarrow \mathbf{p} \text{ is activated.}$$

Furthermore, if the path $\mathbf{p}$ is activated,

$$\text{for any } h \in \mathbf{p},\ \phi(h) = 1$$
$$\Rightarrow \prod_{h \in \mathbf{p}} \phi(h) = 1$$
$$\Rightarrow f^{\mathbf{p}}(X) = (W^{\mathbf{p}} X + b^{\mathbf{p}}) \prod_{h \in \mathbf{p}} \phi(h) \text{ is non-zero if } (W^{\mathbf{p}} X + b^{\mathbf{p}}) \text{ is non-zero.}$$

□

A.2 Proof of Proposition 2

For a multi-way complete path $\mathbf{p}$ for a ReLU NN, we define $W^{\mathbf{p}}$ and $b^{\mathbf{p}}$ by summing up the weights and biases of all possible one-way complete paths that are subsets of the multi-way complete path $\mathbf{p}$.

$$W^{\mathbf{p}} = \sum_{\substack{p \text{ is a one-way complete path} \\ \text{that is a subset of } \mathbf{p}}} W^{p} \tag{iii}$$

$$b^{\mathbf{p}} = \sum_{\substack{p \text{ is a one-way complete path} \\ \text{that is a subset of } \mathbf{p}}} b^{p}. \tag{iv}$$

We define the *piecewise linear model* $f^{\mathbf{p}}(X)$ *for the path* $\mathbf{p}$ as

$$f^{\mathbf{p}}(X) = (W^{\mathbf{p}} X + b^{\mathbf{p}}) \prod_{h \in \mathbf{p}} \phi(h). \tag{v}$$

Proposition 2
For a multi-way complete path $\mathbf{p}$*, the piecewise linear model* $f^{\mathbf{p}}(X)$ *represents a summation of terms from Equation* (2)*, and is non-zero if and only if the path is activated and* $(W^{\mathbf{p}} X + b^{\mathbf{p}})$ *is non-zero.*

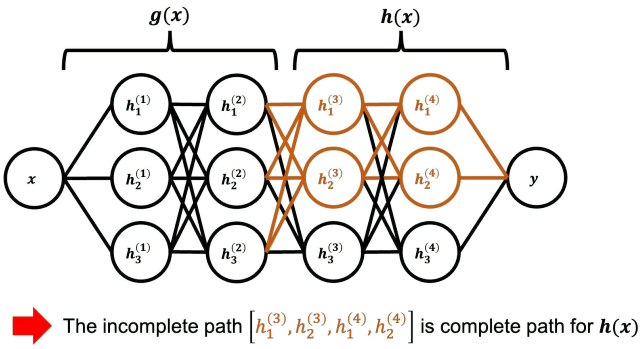


Figure A: Example of incomplete path with *depth* 2

(a) (b) (c) (d) (e)

Figure B: An example of capturing multiple features through the paths. (a) *fox* class input; (b) an explanation by the path using the *fox*'s eyes for the *fox* class prediction; and (c) using the *fox*'s ear. (d, e) attribution maps for (b) and (c)

Proof. In a multi-way complete path, there exist hidden units from the same hidden layer. Without loss of generality, let h_i and h_j be the hidden units in the multi-way complete path from same hidden layer. From Equation (1), we can substitute $\phi(h_i)$ as $\phi(h_i)\phi(h_j)+\phi(h_i)\phi(-h_j)$ and $\phi(h_j)$ as $\phi(h_j)\phi(h_i)+\phi(h_j)\phi(-h_i)$ because $\phi(a)+\phi(-a)=1$. In other words, the decomposition from $\phi(h_i)$ to $\phi(h_i)\phi(h_j)$ and $\phi(h_i)\phi(-h_j)$ means that the $\phi(h_i)\phi(h_j)$ indicates both h_i and h_j are activated, while $\phi(h_i)\phi(-h_j)$ indicates only h_i is activated. In both cases, h_i is activated ($\phi(h_i)$).

$$\begin{aligned} & W_i\phi(h_i) + W_j\phi(h_j) \\ &= W_i\phi(h_i)\,(\phi(h_j)+\phi(-h_j)) + W_j\phi(h_j)\,(\phi(h_i)+\phi(-h_i)) \\ &= (W_i + W_j)\phi(h_i)\phi(h_j) + \alpha(h_i, h_j), \end{aligned}$$

where $\alpha(h_i, h_j)$ is the rest of the term except $(W_i + W_j)\phi(h_i)\phi(h_j)$. □

B Experiment Environment

All experiments were conducted under uniform computing conditions, leveraging a single Quadro RTX 6000 GPU, running on Ubuntu 18.04, with Cuda 10.2 and Pytorch 1.11.0.

C Baseline Codes Used

- We used the **Captum**[2] package and **PAIR Saliency**[3] package for **Saliency**, **Input x Gradient**, **Integrated Gradients**, **Guided Backpropagation**, **Guided GradCAM**, and **Blur Integrated Gradients**.
- We used the **RISE** (Petsiuk *et al.*, 2018)[4] code for the insertion and deletion game (Section 5.3 in the main script).
- We used the training code[5] for the model, as introduced in Section G, on CIFAR10.

[2]https://captum.ai
[3]https://pair-code.github.io/saliency
[4]https://github.com/eclique/RISE/tree/master
[5]https://github.com/kuangliu/pytorch-cifar

(a) (b) (c) (d) (e)

Figure C: An example of capturing multiple objects through the paths. (a) *husky* class input (b) The path that explains the *husky* class prediction using the left husky and (c) right husky. (d, e) attribution maps for (b) and (c)

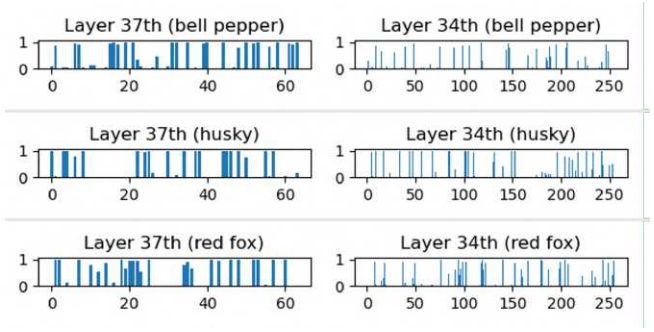


Figure D: The frequency of hidden units for each layer included in paths per class when paths are respectively configured for all data on the curated subset of the ImageNet dataset. The x-axis represents the index of hidden units in that layer, and the y-axis represents the frequency ratio at which units of that index are selected. For better visualization, only the 37^{th} and 34^{th} layers are provided, and only three classes (*bell pepper*, *husky*, and *red fox*) are provided.

D Supplementary Material for Section 5.1

As previously discussed in Section 5.1, our pathwise explanation is capable of generating diverse explanations for a single input by considering multiple paths. In this supplementary section, we aim to further elucidate the content of Section 5.1 by presenting attribution maps for the decomposed explanations of both multiple features and multiple objects. Figure B illustrates the generated explanations for an input characterized by multiple features, while Figure C depicts the explanations for an input containing several objects.

The algorithm for multiple path construction assumes that neurons in the higher layers have high-level features like the fox's eye. We generated multiple paths that include each hidden neuron extracted in Algorithm 1 from the top layer. For example, if a $path^{(N)}=\{h_1, h_2\}$ is generated at step $n=N$ in Algorithm 1, we assumed $path^{(N)}=\{h_1\}$ and proceeded to the next step to create a path. We also assumed $path^{(N)}=\{h_2\}$ and proceeded to the next step to create another path.

E Common Path Per Class

This section describes the consistent explainability of the proposed method by revealing the presence of frequently selected hidden units for each class, i.e., a common path for every class. As depicted in Figure D, the frequency with which hidden units are selected for path configuration is notably polarized (either extremely low or high) across all data points for each class in the dataset. This observation suggests that, for every layer and class, certain specific indices are chosen with high frequency, while others are rarely selected or not at all. Moreover, regardless of whether the classes are visually distinct (e.g., *bell pepper* and *husky*) or bear visual similarities (e.g., *husky* and *red fox*), the frequently selected hidden units differ significantly. In other words, a common path is uniquely associated with a specific class. Consequently, these findings underscore the capability of our pathwise explanation method to reveal the presence of a distinct path integral to the decision-making process for each class. This further bolsters the argument that our proposed method, which leverages only a subset of hidden units, offers a more consistent explanation compared to other methods that utilize all units.

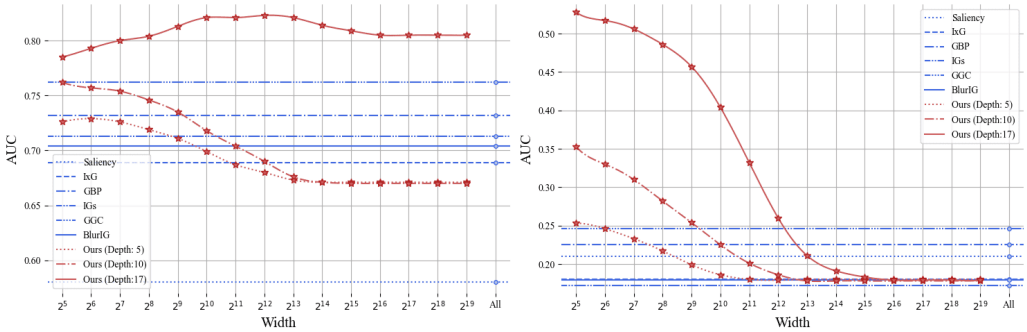


Figure E: Different Architecture (ResNet) Experiment: Quantitative explanation via *insertion* and *deletion* depending on the maximum number of units per layer (*width*) and the number of layers (*depth*) for path configuration. In our method, a *depth* of 17, which is the maximum, represents a complete path, while any lesser value indicates an incomplete path.

	Saliency	IxG	IGs	GBP	GGC	BlurIG	Ours
Insertion (↑)	0.580	0.689	0.713	0.732	0.762	0.704	**0.805**
Deletion (↓)	0.210	0.180	**0.172**	0.225	0.246	0.181	0.179

Table A: Different Architecture (ResNet) Experiment: Area Under Curve (AUC) of *Insertion* and *Deletion*. ↑ indicates that the larger the value, the better the explanation, whereas ↓ indicates the opposite.

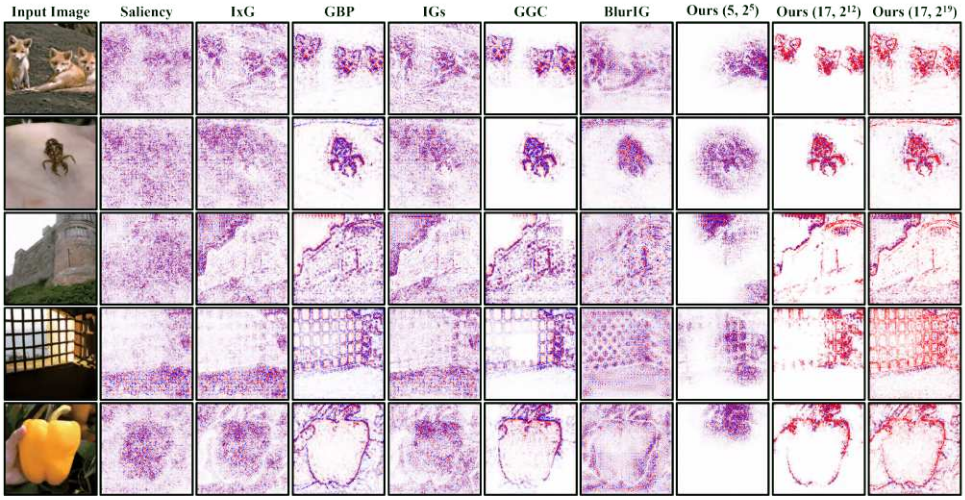


Figure F: Different Architecture (ResNet) Experiment: Qualitative explanation depending on the maximum number of units per layer (*width*) and the number of layers (*depth*) for path configuration. The expression enclosed in parentheses in the proposed method is as follows: (*depth*, *width*). Positive and negative attributions are indicated in red and blue.

F Further Comparison via a Different Architecture: ResNet

Our pathwise method, in addition to the VGG architecture as discussed in the main script, can also be utilized to explain the decision-making process within another well-known architecture, ResNet. Unlike VGG, ResNet contains the residual connection, which operates based on the summation. However, when the path through all layers is modeled, it becomes apparent that the piecewise linear model for the path in ResNet is identical to that in VGG. In fact, by unfolding the formula as in Equation 2 and then extracting terms with ϕ for all hidden units as in Equation 3, this can be revealed.

Table A, Figure E, and Figure F show the results of experiments with ResNet-18 (He *et al.*, 2016), using the same settings as Table 1, Figure 8, and Figure 9 in the main script. Consequently, our method surpasses the others in the insertion metric and provides competitive performance in the deletion metric, even within the ResNet architecture. Furthermore, our study illustrates that the trends in path configuration, concerning changes in *width* and *depth*, align with those observed in VGG.

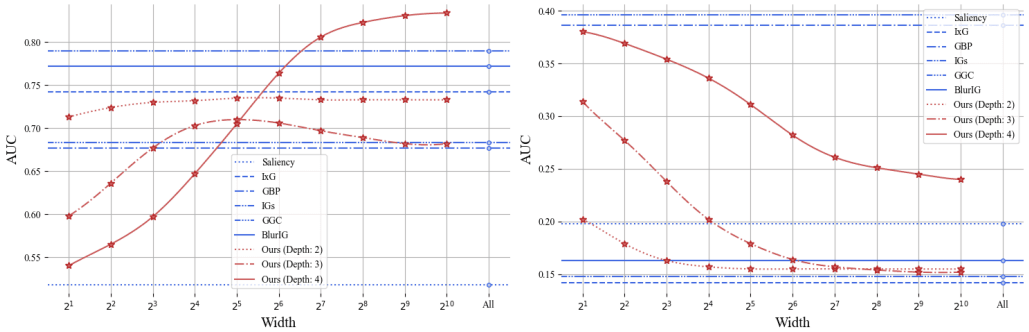


Figure G: Low-Resolution Classification (CIFAR10) Experiment: Quantitative explanation via *insertion* and *deletion* depending on the maximum number of units per layer (*width*) and the number of layers (*depth*) for path configuration. In our method, a *depth* of 4, which is the maximum, represents a complete path, while any lesser value indicates an incomplete path

	Saliency	IxG	IGs	GBP	GGC	BlurIG	Ours
Insertion (↑)	0.518	0.742	0.790	0.677	0.683	0.772	**0.833**
Deletion (↓)	0.198	**0.142**	0.148	0.386	0.396	0.163	0.239

Table B: Low-Resolution Classification (CIFAR10) Experiment: Area Under Curve (AUC) of *Insertion* and *Deletion*. ↑ indicates that the larger the value, the better the explanation, whereas ↓ indicates the opposite.

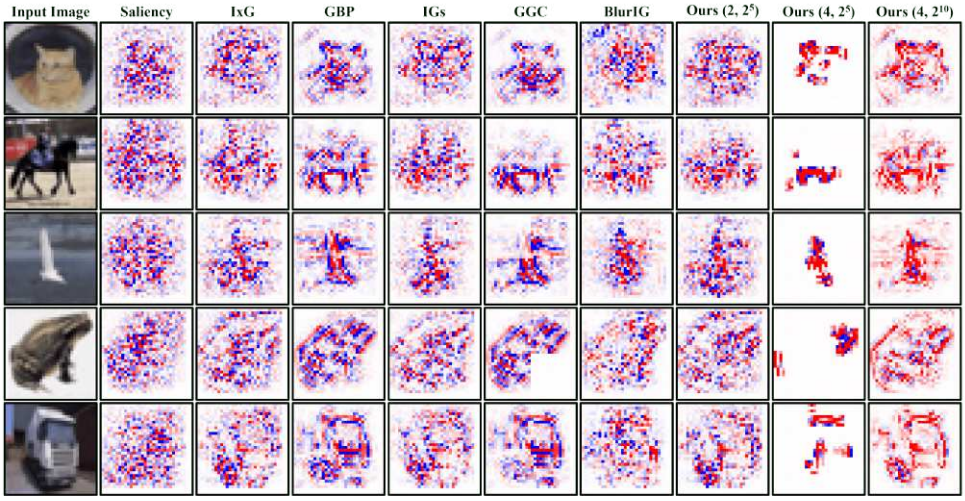


Figure H: Low-Resolution Classification (CIFAR10) Experiment: Qualitative explanation depending on the maximum number of units per layer (*width*) and the number of layers (*depth*) for path configuration. The expression enclosed in parentheses in the proposed method is as follows: (*depth*, *width*). Positive and negative attributions are indicated in red and blue.

G Further Comparison in Low-Resolution Classification: CIFAR10

This section provides additional quantitative comparisons performed on the CIFAR10 dataset (Krizhevsky *et al.*, 2009). These comparisons were omitted from the main manuscript due to space constraints. CIFAR10 is a low-resolution dataset comprising 10 distinct classes. For our experiments on this dataset, we employed a toy model with three convolutional layers (each with a kernel size of 3) and two fully connected layers, offering an intuitive explanation suitable for its small resolution.

Even on CIFAR10, our method surpasses other approaches in terms of the *insertion* metric, as evidenced in Table B. Moreover, Figures G and H indicate that the proposed pathwise method delivers both quantitatively and qualitatively appropriate explanations for the CIFAR10 dataset. Specifically, as depicted in Figure G, larger *width* and *depth* values can improve the upper bound of the *insertion* metric, echoing the observations made in the main manuscript. However, our explanations for the *deletion* metric show a slightly different trend when low-level layers are incorporated into the path due to a high *depth* value. This deviation arises because the object-to-image size ratio in this dataset is substantial, given the image's very low resolution. Contrary to the *insertion* metric, where pixels are inserted into a blank image factoring in the image's local area via low-level layers, in the *deletion* metric, surrounding pixels can influence the model's decision when

pixels are removed from the original image, considering the image's local area through low-level layers. Nonetheless, it's worth noting that our method still demonstrates competitive performance in the *deletion* metric when adjusting the *depth*. Figure H visually showcases our attribution maps, influenced by both *depth* and *width*. Due to the image's limited resolution, attributions cover the entire area when the *depth* is minimal. Yet, it's evident that essential features remain prominently highlighted when the *depth* is maximized, especially with appropriate *width* adjustments.