Machine Learning (2026) 115:17
https://doi.org/10.1007/s10994-025-06957-0

Mechanistic Interpretability of ReLU Neural Networks Through Piecewise-Affine Mapping

Arnab Barua[1] · **Mobyen Uddin Ahmed**[1] · **Shahina Begum**[1]

Received: 29 July 2025 / Revised: 8 December 2025 / Accepted: 12 December 2025

Abstract

Rectified linear unit (ReLU) based neural networks (NNs) are recognised for their remarkable accuracy. However, the decision-making processes of these networks are often complex and difficult to understand. This complexity can lead to challenges in error identification, establishing trust, and conducting thorough analyses. Existing methods often fail to provide clear insights into the actual computations occurring within each layer of these networks. To address this challenge, this study introduces a mechanistic interpretability method called ReLU Region Reasoning (Re3). This method uses the known piecewise-linear characteristics of ReLU networks to offer insights into neuron activation and accurately assess how each feature contributes to the final output and probability. Re3 effectively determines neuron activations and evaluates the contribution of each feature within a specified linear region. Experiments conducted on multiple benchmark datasets, including both tabular and image data, demonstrate that Re3 can replicate individual predictions without error, align feature importance with domain expertise, and maintain consistency with current explanatory methods, thereby avoiding the typical randomness. Analysing neurons reveals activation sparsity and identifies dominant units, thus providing clear targets for model simplification and troubleshooting. By ensuring transparency and algebraic accessibility in each stage of a ReLU-based NN's decision process, Re3 can be a valuable practical tool for achieving precise mechanistic interpretability.

Keywords Neural networks · Rectified linear unit · Reasoning · Interpretability

Mobyen Uddin Ahmed and Shahina Begum contributed equally to this work.

Editor: Panagiotis Papapetrou.

✉ Arnab Barua
arnab.barua@mdu.se

Mobyen Uddin Ahmed
mobyen.uddin.ahmed@mdu.se

Shahina Begum
shahina.begum@mdu.se

[1] School of Innovation, Design and Engineering, Mälardalen University, Universitetsplan 1, 72220 Västerås, Västmanland, Sweden

1 Introduction

Neural networks (NNs) exhibit state-of-the-art performance in various applications, including computer vision, language processing, and scientific research. However, their internal decision-making processes remain difficult to understand. These models are often described as black boxes, as even a single forward pass involves thousands or millions of hidden units interacting through complex, non-linear transformations (Guidotti et al., 2018). This lack of transparency undermines trust, particularly in safety-critical domains such as healthcare and finance, where stakeholders require clear explanations for a model's classifications or predictions. Furthermore, without transparency, diagnosing failures, detecting biases, and ensuring regulatory compliance become significantly challenging. Consequently, there is a growing consensus on the need to develop methods that enable the exploration of NNs and provide human interpretable explanations for their decisions (Goodfellow et al., 2016).

Existing interpretability approaches for NNs can be divided into post-hoc surrogates and exact model introspection (Lipton, 2018; Madsen et al., 2022; Velmurugan et al., 2023). Post-hoc methods, like Local Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016) and SHapley Additive exPlanations (SHAP) (Lundberg & Lee, 2017), create simpler and interpretable models, usually linear regressions (LR) or decision trees (DT) based on small changes in the input data (Salih et al., 2025). They treat the NNs as a black box (Stadlhofer & Mezhuyev, 2023). While these methods are flexible and can work with different types of models, they only provide approximations of the actual decision boundary. Their explanations can also vary significantly depending on random factors such as sampling noise, the choice of kernel bandwidth, or the random seed. Again, exact-extraction techniques such as OpenBox (Chu et al., 2018) and TropEx (Trimmel et al., 2021) take advantage of the fact that a ReLU network is a piecewise-affine function. They aim to extract every linear segment where the network behaves exactly linearly. However, the number of these segments increases exponentially as the network gets deeper and wider, making simple extraction impractical for anything but very small networks.

While LIME and SHAP offer intuitive approximations of model behaviour, they often obscure the true internal logic of NNs. Conversely, exact interpretation techniques for ReLU networks can expose the underlying piecewise-linear structure, but they typically lack per-sample attribution and are not easily integrated into standard development workflows. These methods, often developed for formal verification, can be computationally intensive, sometimes requiring days to analyse even small networks, making them impractical for real-time or developer-facing applications (Bak et al., 2020). These limitations highlight a fundamental need for a method that not only reveals the predictions made by the model but also explains how and why those predictions are reached, down to the level of individual neurons and features. To address these gaps, a set of guiding questions was formulated to shape the design of the proposed approach. These questions reflect the practical needs of developers and researchers working with NN systems in real-world settings.

- **How can the internal processes of NNs be revealed and explored?** Understanding how a network's neurons activate and interact is crucial for ensuring transparency and trust in safety-critical applications.
- **How do NNs create decision regions?** NNs form linear decision regions in the input space through layer-wise transformations. Analysing region formation, training-testing

similarity, sample distribution, and stability can improve reliability.

- **How do neurons get activated and interact within the hidden layers?** Neuron activation patterns indicate how features are abstracted and combined. Identifying sparsity, redundancy, or specialisation among neurons can enhance pruning strategies and architectural improvements (Montavon et al., 2018).
- **How do features move from the input through the hidden layers to the final decision of the network?** Tracing information flow across layers enables precise attribution and deeper insights into model predictions and their causes (Samek et al., 2021).

The goal of this study is to develop a practical tool for understanding how ReLU-based NNs function. By taking advantage of the piecewise-linear nature of these networks, the tool will identify inactive or minimally changing neurons and discover linear regions where each behaves like a simple decision rule. Within each of these regions, it will be possible to calculate precisely how each feature affects both the raw class score and the softmax probability, without approximations. By combining these individual contributions across various test samples, a consistent and reliable ranking of feature importance will be generated. Experiments conducted on several benchmark datasets will show that this straightforward, piecewise-linear approach accurately reproduces every prediction and produces feature rankings that align with established knowledge, without the unpredictability or errors associated with black-box explainers. Based on this foundation, the study offers six key contributions:

- **Proposed Interpretable Method:** Introduces Re3, an efficient approach that calculates the exact per-sample affine mapping of any trained ReLU network, revealing its internal computations.
- **Practical, Developer-Oriented Pipeline:** A lightweight toolkit combining NumPy and PyTorch makes it easy to handle training, analytic forward passes, and visualisation all in one seamless workflow.
- **Local and Global Insights:** Exact probability contributions illustrate how individual features influence specific predictions, and these contributions are aggregated across the dataset to produce stable global feature rankings.
- **Empirical Validation and Comparison:** The method was evaluated using benchmark datasets that consist of tabular and image data, leading to quantitative comparisons. This systematic validation instils confidence that the method's precise explanations align with established theories.
- **Linear Region Analysis:** This method counts both active and inactive neurons, identifying the linear region. It helps in understanding region characteristics and provides insights for model improvement.
- **Exact What-If Analysis:** Calculating the exact per-sample affine mapping, the method offers a precise linear mapping for each input. This representation enables true *what-if* analysis, allowing developers to algebraically adjust any feature value and instantly calculate the precise change in the model's output without any approximations.

The organisation of this article is as follows. Section 2 presents related work on network interpretability. Section 3 outlines the methodology used in this study. Section 4 provides extensive experiments conducted on five benchmark datasets, including measures of accuracy, explanation quality, and statistical validation. Section 5 discusses the findings in relation to the guiding questions and contributions. Finally, Sect. 6 summarises the article and proposes directions for future research.

2 Related Works

In machine learning (ML), interpretability methods are generally categorised into different groups (Murdoch et al., 2019). The authors of the article (Doshi-Velez & Kim, 2017) differentiate between methods that incorporate transparency directly into the model and those that provide explanations for a trained black box. In the article Lipton (2018), approaches are further divided into transparent models where the entire reasoning process can be scrutinised and post-hoc methods, which analyse predictions after they have been made. Guidotti et al. (2018) classify explanations as either model-agnostic or model-specific, and discuss the trade-offs associated with each type. More recently, Islam et al. (2022) performed a systematic review of explainable AI methods across various application domains and tasks, emphasising the increasing significance of domain-specific interpretability and the necessity for robust evaluation metrics.

Post-hoc techniques are designed to provide local approximations of complex models. For instance, LIME (Ribeiro et al., 2016) uses simple linear regressions to explain individual predictions, while SHAP (Lundberg & Lee, 2017) leverages Shapley values to fairly attribute contributions from various features. On the other hand, gradient-based methods like Integrated Gradients (IG) (Sundararajan et al., 2017) and Layer-wise Relevance Propagation (LRP) (Bach et al., 2015) work by tracing importance back through the layers of the network. Although these approaches are versatile and applicable across different models, they can introduce errors due to sampling or approximations.

ReLU networks break down their input space into various regions, each defined by a straightforward linear function of the form $f(x) = A\,x + D$ (Goodfellow et al., 2016; Montufar et al., 2014; Raghu et al., 2017). Tools like OpenBox (Chu et al., 2018) and TropEx (Trimmel et al., 2021) take advantage of this approach to identify all the different linear sections. However, they often struggle with a huge number of potential regions as the networks get larger. Recent studies, including the work by Berzins (2023), suggest using edge-based subdivision to reduce unnecessary overlap during the extraction process.

Several studies focus on simplifying NNs to make them easier to understand. For example, Neuron Importance Score Propagation (NISP) examines the back-propagated gradients to rank hidden units (Yu et al., 2018), while Liu et al. (2017) introduced Network Slimming, which creates sparsity in channels using learned scaling factors. Hooker et al. (2019) investigate how different pruning methods impact the stability of explanations after the fact. Meanwhile, Molchanov et al. (2016) suggest a method for ranking and removing convolutional filters based on a second-order Taylor expansion of the loss, aiming to improve efficiency during inference.

Recent studies are breaking down neural circuits into parts that are easier to understand. For example, Olah et al. (2018) examined features of individual neurons and layers in vision models. Nanda et al. (2023) developed a system that automatically discovers smaller circuits in transformer models. In contrast, the proposed method distinguishes itself from post-hoc approaches like LIME, SHAP, Integrated Gradients, and LRP, which rely on sampling or gradient techniques to interpret a network's behaviour. Instead, it provides precise attributions for each feature by directly simplifying the complex areas of the network. Unlike OpenBox and TropEx, which can be costly because they analyse every individual linear region, this approach employs weights of training and identifies inactive neurons. This strategy reduces the number of regions while still ensuring accurate results.

While other pruning methods, such as NISP, Network Slimming, and Taylor-based filter pruning, focus on aspects like reducing the network size or stabilising explanations, this technique centres on how each neuron learns. This ensures that only the truly useful neurons are kept for explanations. Finally, while many large-scale efforts break down entire transformer models, this approach applies similar ideas to smaller ReLU networks, providing clear, mathematical contributions for each feature.

3 Methodology

This section outlines the methodology of this study, which consists of several steps, from dataset to evaluation, as illustrated in Fig. 1.

3.1 Dataset

In this study, five benchmark datasets from the UCI Machine Learning Repository[1] were utilised to evaluate the proposed methodology. The well-known *Iris* dataset presents a small, well-understood multiclass problem, consisting of 150 samples characterised by four parameters: *sepal length*, *sepal width*, *petal length*, and *petal width*. These samples belong to three species: *setosa*, *versicolor*, and *virginica*. Similarly, the *Seeds* dataset comprises 210 wheat kernel samples, each described by seven geometric features: *area*, *perimeter*, *compactness*, *kernel length*, *kernel width*, *asymmetry coefficient*, and *groove length*. This dataset is divided into three varieties: *kama*, *rosa*, and *canadian*.

To evaluate scalability and binary classification performance, three larger datasets were used. *The Accelerometer Gyro Mobile Phone* (AGMP) dataset encompasses 31,991 records from six sensor readings: *accX*, *accY*, *accZ*, *gyroX*, *gyroY*, and *gyroZ*. Each record is labelled as either *standing* or *walking*, presenting a time-series classification challenge. The *CDC Diabetes Health Indicators* (CDCDHI) dataset contains 253,680 health survey entries with 21 recorded variables, of which 16 were selected for modelling. These variables include *HighBP*, *HighChol*, *CholCheck*, *BMI*, *Smoker*, *Stroke*, *HeartDiseaseorAttack*, *PhysActivity*, *Fruits*, *Veggies*, *HvyAlcoholConsump*, *AnyHealthcare*, *GenHlth*, *MentHlth*, *PhysHlth*, and *DiffWalk*, which are used to classify individuals as having diabetes or not having diabetes. Lastly, the *Spambase* dataset features 4,601 samples characterised by 57 continuous attributes, classified as either *spam* or *not spam*. Due to the large number of features, the complete list is omitted here. Collectively, these datasets encompass a variety of sizes, feature dimensionalities, and class complexities, providing a solid foundation for assessing both network behaviour and explanation fidelity.

This study demonstrates the applicability of Re3 beyond tabular datasets by using the MNIST and Fashion-MNIST datasets for a convolutional neural network (CNN) architecture. Both datasets contain 70,000 grayscale images with dimensions of 28“× 28 pixels

[1] https://archive.ics.uci.edu/ml/index.php.

and consist of 10 classes each. The MNIST dataset includes handwritten digits, while the Fashion-MNIST dataset features clothing items.

3.2 Neural Network (NN)

NN is a computational model that draws inspiration from the way the human brain functions (Agatonovic-Kustrin & Beresford, 2000). It is composed of layers of units called neurons that connect with one another. These neurons process incoming signals by using weighted connections and activation functions. The learning process happens through a backpropagation where the model fine-tunes the connection weights based on the difference between what the network predicts and the actual outcomes (Goodfellow et al., 2016; Rumelhart et al., 1986). More information about it can be found in Goodfellow et al. (2016) and Agatonovic-Kustrin and Beresford (2000). This study evaluates three simple NN architectures with increasing depth to explore the impact of layer size and network depth on neuron activations. The input layer is designed to correspond with the number of features in the dataset, while all hidden layers use the rectified linear unit (ReLU) activation function. For classification purposes, the softmax function is applied in the output layer, which is customised to accommodate the number of target classes.

The first and simplest model consists of a single hidden layer with 8 neurons. The second model introduces an additional hidden layer that contains 4 neurons. The final model employs three hidden layers with neuron configurations of 16, 8, and 4, respectively. Through this structured approach, the study aims to provide insights into how variations in architecture influence performance in classification tasks.

3.3 ReLU Region Reason (Re3)

This work proposes the ReLU Region Reason (Re3) method for identifying the active ReLU region of a trained NNs and collapsing it into a single linear function. Each ReLU hidden unit segments its input space into two half-spaces: *active* and *inactive*. Here, *active* refers to instances when its pre-activation is positive, and *inactive* otherwise. For a given input sample x, the combination of on/off patterns across all units is represented by a binary vector r. Within the corresponding region, the network behaves as an affine function defined by:

$$f(x) = A_r x + D_r, \tag{1}$$

where (Eq. 1), A_r denotes the collapsed weight matrix obtained by masking out inactive neurons in each layer's weights, while D_r represents the collapsed bias vector formed by combining the original biases under the same mask. By directly computing A_r and D_r, the proposed method reveals the precise linear rule that the network uses for the specific input. This accurate description illustrates how each feature traverses the active neurons to generate the final output.

3.3.1 Piecewise-Affine of ReLU Networks

To demonstrate how ReLU activations divide the input space based on each sample's affine form in Eq. 1, it is essential to show that the network behaves like a simple linear map in specific regions.

A function $f : \mathbb{R}^d \to \mathbb{R}^k$ is called piecewise-affine if its input space can be divided into a finite number of convex regions, where f behaves like an affine function in each region. In a ReLU network, the only source of nonlinearity is the activation function $\max(0, z)$. Consequently, although the network as a whole is non-linear, it behaves like a single linear function, as shown in Eq. 1, whenever the input x remains within one of these regions. To understand how these regions are formed, the next step is to analyse how each hidden unit divides its input space.

Consider each neuron i in layer l that computes a pre-activation defined by the equation:

$$z_i^{(l)} = w_{l,i}^\top x^{(l-1)} + b_{l,i} \tag{2}$$

where $x^{(0)} = x$. The value $z_i^{(l)}$ can be compared to zero, which defines a hyperplane described by:

$$\{x : w_{l,i}^\top x^{(l-1)} + b_{l,i} = 0\} \tag{3}$$

This hyperplane divides its input into an *active* half-space where $z_i^{(l)} > 0$ and an *inactive* half-space where $z_i^{(l)} \leq 0$. Collecting these *active* and *inactive* tests for all n_l neurons in layer l return a binary mask:

$$S^{(l)} = \mathbf{1}\big(W_l\, x^{(l-1)} + b_l > 0\big) \tag{4}$$

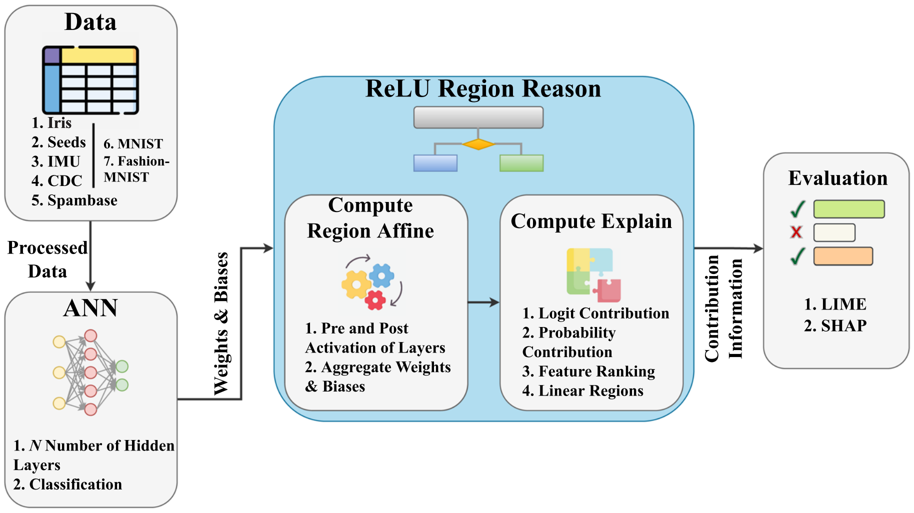


Fig. 1 Overview of the methodological pipeline

where $\mathbf{1}(\cdot)$ is the indicator function that returns 1 for active neurons and 0 for inactive ones. The complete set of masks $\{S^{(1)}, \dots, S^{(L-1)}\}$ uniquely labels a region r of the input space $\mathbb{R}^d$. With these masks established, it is possible to collapse the entire network into a single affine map that is valid within region r. Inside region r, every inactive neuron contributes nothing to the computation. By inserting a diagonal matrix $\mathrm{diag}(S^{(l)})$ after each weight matrix W_l, all inactive paths are automatically excluded from the calculations. Therefore, by chaining these masked matrices and summing the corresponding biases, it is possible to produce exactly the single affine mapping introduced in Eq. 1 for every x in region r.

An illustration of the process is provided by a NN with two hidden layers, as described below. Let's consider an input $x \in \mathbb{R}^n$ and denote the parameters of the first hidden layer as follows:

$$W^{(1)} \in \mathbb{R}^{n_1 \times n}, \quad b^{(1)} \in \mathbb{R}^{n_1} \tag{5}$$

For the second hidden layer, the parameters are:

$$W^{(2)} \in \mathbb{R}^{n_2 \times n_1}, \quad b^{(2)} \in \mathbb{R}^{n_2} \tag{6}$$

The output layer is defined by:

$$W^{(3)} \in \mathbb{R}^{k \times n_2}, \quad b^{(3)} \in \mathbb{R}^{k} \tag{7}$$

For a given input x, computes the pre-activation of the first layer:

$$Z^{(1)} = W^{(1)}\, x + b^{(1)} \tag{8}$$

Then apply the ReLU activation function to $Z^{(1)}$ to produce $H^{(1)}$:

$$H^{(1)} = \max\big(0,\ Z^{(1)}\big), \tag{9}$$

This can also be expressed using the mask $S^{(1)} = \mathbf{1}(Z^{(1)} > 0)$ as:

$$H^{(1)} = S^{(1)} \odot Z^{(1)} \tag{10}$$

Next, proceed to the second layer, where the computation of the pre-activation is as follows:

$$Z^{(2)} = W^{(2)}\, H^{(1)} + b^{(2)}, \quad H^{(2)} = \max\big(0,\ Z^{(2)}\big) = S^{(2)} \odot Z^{(2)}, \tag{11}$$

with $S^{(2)} = \mathbf{1}(Z^{(2)} > 0)$. Finally, the output of the network is given by the affine transformation:

$$\mathrm{output} = W^{(3)}\, H^{(2)} + b^{(3)}. \tag{12}$$

Knowing the binary masks $S^{(1)}$ and $S^{(2)}$ identifies the region *r* to which *x* belongs. Inserting $\mathrm{diag}(S^{(1)})$ and $\mathrm{diag}(S^{(2)})$ in between the weight matrices zeroes out any inactive neurons, allowing all three layers to be collapsed into the single affine mapping of Eq. 1.

$$A_r = W^{(3)}\,\mathrm{diag}(S^{(2)})\,W^{(2)}\,\mathrm{diag}(S^{(1)})\,W^{(1)} \tag{13}$$

$$D_r = b^{(3)} + W^{(3)}\,\mathrm{diag}(S^{(2)})\,b^{(2)} + W^{(3)}\,\mathrm{diag}(S^{(2)})\,W^{(2)}\,\mathrm{diag}(S^{(1)})\,b^{(1)}. \tag{14}$$

Here, A_r represents all active weight pathways from input to output, while D_r accumulates the corresponding biases. Thus, for every x in region r, the network's computation is given by $f(x) = A_r x + D_r$.

In an L-layer NN, a similar approach can be applied. Specifically, a diagonal mask, represented as $\mathrm{diag}(S^{(l)})$, is inserted between each pair of weight matrices to control the flow of information between layers. After applying the mask, matrix multiplication is performed to create the output denoted as A_r. Additionally, bias terms are incorporated into a combined term known as D_r. The computational cost associated with this process scales linearly with the number of layers, as each layer requires only one mask application and one matrix multiplication. The theoretical foundation of the approach is formally established in Sect. 3.3.2.

3.3.2 Formal Justification

To establish the mathematical basis of Re3, a formal proof demonstrating the piecewise-affine property of ReLU networks is presented below.

Proposition 1 *Let $f : \mathbb{R}^d \to \mathbb{R}^C$ be a feed-forward NN composed of affine transformations and ReLU activations. The input space $\mathbb{R}^d$ can be divided into a finite collection of regions $\{R\}$, each characterised by a unique activation pattern of the hidden units, such that for every $x \in R$ the network reduces to an affine map*

$$f(x) = A_R x + D_R,$$

where $A_R \in \mathbb{R}^{C\times d}$ and $D_R \in \mathbb{R}^C$ depend only on the region R.

Proof Consider a single hidden layer with weights $W^{(1)} \in \mathbb{R}^{m\times d}$ and bias $b^{(1)} \in \mathbb{R}^m$. The pre-activation is given by,

$$Z^{(1)} = W^{(1)}x + b^{(1)}.$$

The activation mask $S^{(1)} \in \{0,1\}^m$ is defined with entries $S_i^{(1)} = \nVdash_{Z_i^{(1)}>0}$, where $\nVdash$ denotes the indicator function. Applying the ReLU activation yields,

$$H^{(1)} = \sigma(Z^{(1)}) = \mathrm{diag}(S^{(1)})(W^{(1)}x + b^{(1)}).$$

For a fixed mask $S^{(1)}$, this mapping is affine in *x*.

For a second hidden layer with weights $W^{(2)} \in \mathbb{R}^{k\times m}$ and bias $b^{(2)} \in \mathbb{R}^k$, the output becomes,

$$H^{(2)} = \sigma(W^{(2)}H^{(1)} + b^{(2)}) = \mathrm{diag}(S^{(2)})\big(W^{(2)}\ \mathrm{diag}(S^{(1)})(W^{(1)}x + b^{(1)}) + b^{(2)}\big),$$

where $S^{(2)}$ denotes the activation mask of the second layer. Again, for fixed masks $(S^{(1)}, S^{(2)})$, this mapping remains affine in x.

The argument extends by induction: if the composition of the first $(\ell - 1)$ layers is affine under fixed masks, then adding the ℓ-th layer (an affine transformation followed by a fixed ReLU mask) preserves this affine structure. Thus, for any network of finite depth, the output is affine in x whenever all activation masks remain fixed.

The collection of masks across all layers uniquely determines a region R. Within each such region, the network can be expressed as,

$$f(x) = A_R x + D_R,$$

where A_R and D_R result from composing the masked affine transformations layer by layer. Since the network contains finitely many neurons, the number of distinct mask configurations and hence the number of regions is finite.

Partition of the input space. The regions $\{R\}$ form a partition of the entire input space $\mathbb{R}^d$. Each region R corresponds to a fixed activation pattern across all hidden layers. Formally, for a network with L layers and n_ℓ neurons in layer ℓ, a region R is characterized by a tuple of masks $(S^{(1)}, S^{(2)}, \dots, S^{(L)})$ where $S^{(\ell)} \in \{0, 1\}^{n_\ell}$.

Each region can be described as the set of points satisfying a system of linear inequalities determined by the sign patterns of the pre-activations:

$$R = \{x \in \mathbb{R}^d : \text{sign}(W^{(\ell)} H^{(\ell-1)}(x) + b^{(\ell)}) = S^{(\ell)},\ \forall \ell = 1, \dots, L\},$$

where $H^{(0)}(x) = x$ and $H^{(\ell)}$ denotes the output of layer ℓ. These regions are convex polytopes formed by intersections of half-spaces, and together they tile the input space without overlap. The formal proof that ReLU networks induce such a partition is given in Lemma 2 of Montufar et al. (2014), which establishes that the regions are mutually disjoint and cover $\mathbb{R}^d$..

Universality over datasets. Since $\{R\}$ forms a partition of $\mathbb{R}^d$, any point $x \in \mathbb{R}^d$ lies in exactly one region R. This property is determined entirely by the network architecture and its learned parameters, independent of any particular dataset. Consequently, for any dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, each sample x_i belongs to exactly one region of the partition, and the network's prediction on x_i is given by the corresponding affine map for that region. □

Corollary 1 *Let $f : \mathbb{R}^d \to \mathbb{R}^C$ be a ReLU network with regions $\{R\}$ forming a partition of $\mathbb{R}^d$ and corresponding affine maps $\{(A_R, D_R)\}$. For any dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, each sample x_i lies in exactly one region R_i, and its prediction is*

$$f(x_i) = A_{R_i} x_i + D_{R_i}.$$

3.3.3 Implementation of Algorithm

The proposed method consists of four key components. The first component is Algorithm 1, which serves as the primary driver. This algorithm oversees the entire process by iterating through each sample. Once all samples have been processed, it computes feature ranking frequencies, assesses feature contributions, computes region analysis and generates accuracy and classification reports. The second component is Algorithm 2, which accepts a sample along with the network's weights and biases. It identifies which ReLU units are active and condenses the network into a single affine map. The third component is Algorithm 3, which utilises A_r, D_r, the logits, and the prediction, along with the names of features and classes, to calculate logit and probability contributions. Finally, the fourth component Algorithm 4 generates regions using the region profile from Algorithm 2. After all samples are processed, the explanations are aggregated into a comprehensive table, and model performance metrics are generated.

```
Require: Final weights $(w1, w2, w3)$ and biases $(b1, b2, b3)$ of each layer, $testSamples$
    - Test Samples, $featureNames$ - Feature names, $classNames$ - Class names
Ensure: $featureContribution$ - Sample-wise logit and probability contribution
    of each feature, $featureRanking$ - Summary of sample-wise feature ranking
    $classificationReport$ - Sample-wise classification report
1: Initialize $featureContribution$, $featureRanking$, $classificationReport$ and $allRegions$
2: for $i$ in $testSamples$ do
3:     Initialize $A_r$ - Aggregated affine weights, $D_r$ - Aggregated affine biases, $\ell$ - Logits, and $pred$ - Predictions
4:     $sample \leftarrow testSamples[i]$
5:     $A_r, D_r, \ell, pred, c1, c2, regions \leftarrow$ ComputeRegionAffine($sample, w1, b1, w2, b2, w3, b3$)
6:     $classificationReport \leftarrow pred$
7:     $allRegions \leftarrow regions$
8:     $featureContribution \leftarrow$ ComputeExplain($sample, A_r, D_r, \ell, pred, featureNames, classNames$)
9: end for
10: $regionResults \leftarrow RegionAnalysis(allRegions, testSamples, classificationReport, featureNames, classNames, w1, b1, w2, b2, w3, b3, purityThreshold)$
11: $featureRanking \leftarrow featureContribution$
12: return $featureContribution, featureRanking$, $classificationReport$, $regionResults$
```

Algorithm 1 Re3 algorithm

```
Require: $sample$, $w1, b1, w2, b2, w3, b3$
Ensure: $A_r, D_r, \ell, pred$
1: // Hidden layer 1
2: Compute pre-activation: $Z1 \leftarrow w1\, sample + b1$
3: Build mask: $S1 \leftarrow [Z1 > 0]$
4: Compute post-activation: $H1 \leftarrow \max(Z1, 0)$
5: // Hidden layer 2
6: Compute pre-activation: $Z2 \leftarrow w2\, H1 + b2$
7: Build mask: $S2 \leftarrow [Z2 > 0]$
8: Compute post-activation: $H2 \leftarrow \max(Z2, 0)$
9: // Output layer
10: Computer logits: $\ell \leftarrow w3\, H2 + b3$
11: Compute prediction: $pred \leftarrow \arg\max(\ell)$
12: // Aggregate weights
13: Compute mask inactive weights: $W2^r \leftarrow w2 \circ S1^\top$
14: Compute mask inactive weights: $W3^r \leftarrow w3 \circ S2^\top$
15: $A_r \leftarrow W3^r\,(W2^r\, w1)$
16: // Aggregate biases
17: $D_r \leftarrow b3 + W3^r\, b2 + W3^r\,(W2^r\, b1)$
18: Compute neuron contribution for layer 2: $c2 \leftarrow H2 \odot W3^r[pred]$
19: Compute neuron contribution for layer 1: $c1 \leftarrow H1 \odot W3^r[pred] \cdot W2^r$
20: Compute regions: $regions \leftarrow$ concatenate$(S1, S2)$
21: return $A_r, D_r, \ell, pred, c1, c2, regions$
```

Algorithm 2 ComputeRegionAffine

```
Require: $sample, A_r, D_r, \ell, pred, featureNames$, $classNames$
Ensure: $allLogitProbContribution$ - A DataFrame with index = $featureNames$ and
    columns logit and probability contribution
1: Compute logit-level contributions: $f_{\text{logit}} \leftarrow A_r[pred] \odot sample$
2: Compute Jacobian row for predicted class: $J \leftarrow$ softmaxJacobianRow$(pred)$
3: Compute probability-level contributions: $f_{\text{prob}} \leftarrow J\,(A_r\, x)$
4: $allLogitProbContribution \leftarrow f_{\text{logit}}, f_{\text{prob}}$
5: return $allLogitProbContribution$
```

Algorithm 3 ComputeExplain

```
Require: $allRegions, testSamples, classificationReport, featureNames,$
    $classNames, w1, b1, w2, b2, w3, b3$ $purityThreshold$
Ensure: $regionStatistics$, $Coverage$, $featureProfiles$
1: // Region similarity between train and test
2: Compute train regions: $R_{train} \leftarrow$ set of regions from $allRegions$
3: Compute test regions: $R_{test} \leftarrow$ set of regions from $allRegions$
4: Compute similarity: similarityCount $\leftarrow |R_{train} \cap R_{test}|$
5: Compute didsimilarity: dissimilarityCount $\leftarrow |R_{test} \setminus R_{train}|$
6: // Per-region class purity
7: for each region $r \in R_{test}$ do
8:     $I_r \leftarrow$ indices of samples in region
9:     $y_r \leftarrow y_{test}[I_r]$
10:    Count class frequencies in $y_r$
11:    purity $\leftarrow$ (max class count) / $|I_r|$
12:    isConfusion $\leftarrow$ (purity $<$ $purityThreshold$)
13: end for
14: // Compute coverage metrics
15: Sort regions by size (descending)
16: Computer top 1 coverage: top1 $\leftarrow$ (largest region size) / $N_{test}$
17: Computer top C coverage: topC $\leftarrow$ (sum of $C$ largest regions) / $N_{test}$
18: // Feature importance profiles
19: for each region $r$ in $allRegions$ do
20:    $A_r, D_r \leftarrow$ ComputeRegionAffine($sample$, $w1, b1, w2, b2, w3, b3$)
21:    Compute majority class profile: mean($A_R[c_{maj}] \odot X_r$)
22:    Compute per-class profiles: mean($A_R[c] \odot X_{r,c}$) for each class
23: end for
24: return $regionStatistics$, $Coverage$, $featureProfiles$
```

Algorithm 4 RegionAnalysis

3.4 Evaluation

To assess the accuracy and dependability of the piecewise-affine attributions compared to two well-known explanation methods: LIME and SHAP. Both methods determine feature importance by closely examining the model's decision-making, but they use different approaches to achieve this.

LIME is a popular technique designed to explain the predictions of any black-box ML model in a locally faithful manner. It was introduced by Ribeiro et al. (2016) in a most influential paper *Why Should I Trust You?*. The main idea behind LIME is to build simple, interpretable models, like linear regressions, around individual predictions to mimic the behaviour of the complex model within a small neighbourhood of the input. LIME works by generating perturbed samples around the instance of interest and observing the corresponding changes in predictions. This local surrogate model highlights which features were most influential for that particular decision. More insights about LIME can be found in articles (Christoph, 2020; Garreau & Luxburg, 2020; Guidotti et al., 2018).

SHAP is a well-known method used to explain how ML models make predictions. It was introduced by Lundberg and Lee (2017) and uses ideas from cooperative game theory to determine the importance of each input feature. Essentially, it breaks down a model's predictions to show how much each feature contributes, considering every possible combination of features. One of the great things about SHAP is that it has solid theoretical foundations, which means that it can give reliable explanations that are trustworthy. It works well with various types of models and can provide both specific explanations for individual predictions as well as broader insights for understanding the model as a whole. For further reading on SHAP and its computational strategies, see articles Christoph (2020), Covert et al. (2021) and Kumar et al. (2020).

4 Experiment and Result

4.1 Experimental Setup

Each benchmark dataset, as outlined in Sect. 3.1, was carefully examined for any missing or null values, and none were found. When the target variable was categorical, it was converted into integer class labels to facilitate processing. To ensure that all features were on a comparable scale and to enhance the stability of the training process, continuous inputs were standardised. A *StandardScaler* was applied to the training data, and the same scaling was applied to the test data. The datasets were then divided into train and test sets, with 80% allocated for training and 20% for testing, which helped maintain the class proportions in both groups. No separate validation set was created because the primary objective of these experiments was to analyse the model's behaviour rather than to focus on improving predictive performance. After training in different architecture settings, evaluation on their test sets showed classification accuracies over 80%, with deeper networks achieving better results, as detailed in Online Appendix A.

4.2 Experimental Result on Linear Region Characterization

The piecewise-affine nature of ReLU networks plays a crucial role in characterising their linear regions and reveals distinct clustering patterns within the learned decision structure. Each linear region is associated with a unique activation pattern across the network's neurons, providing a clear framework for understanding the network's behaviour. A detailed analysis of the neuron activation patterns can be found in Online Appendix B. By examining these linear regions, it is possible to gain deeper insights into how the network organises feature space and executes its decision-making process.

4.2.1 Region Discovery Across Networks

Distinct linear regions were identified throughout the training process for each dataset and architecture. The total number of these regions serves as a valuable indicator of the complexity inherent in the learned decision boundary, as well as the partitioning strategy employed by the network. Presented in Table 1 is the number of distinct linear regions identified for each dataset across various layer configurations.

The result reveals notable variation in region counts across different architectures and datasets. For the Iris dataset, single hidden layer networks create 12 regions, while two-layer and three-layer networks produce 17 and 35 regions, respectively. The Seeds dataset

Table 1 The number of linear regions identified by the trained model in three different layer configurations

Dataset	Features	Single hidden layer	Two hidden layes	Three hidden layers
Iris	4	12	17	35
Seeds	7	26	33	52
AGMP	6	66	116	1,226
CDCDHI	16	14	30	1,456
Spambase	57	195	357	1,981

Table 2 The similarity between the training and testing regions, as well as the coverage of samples across various network settings

Dataset	Train region	Test region	Similar	Dissimilar	*Top-1* coverage (%)	*Top-C* coverage (%)
Single hidden layer						
Iris	12	8	8	0	46.67	80.00
Seeds	26	15	13	2	19.05	54.76
AGMP	66	58	56	2	12.56	23.46
CDCDHI	14	14	14	0	66.61	87.78
Spambase	195	139	133	6	10.97	16.72
Two hidden layers						
Iris	17	8	7	1	33.33	70.00
Seeds	33	14	13	1	28.57	59.52
AGMP	116	91	88	3	7.06	13.81
CDCDHI	30	29	29	0	21.57	42.82
Spambase	357	207	171	36	15.64	26.06
Three hidden layers						
Iris	35	11	9	2	26.67	56.67
Seeds	52	25	17	8	16.67	38.10
AGMP	1,226	722	636	86	3.81	7.16
CDCDHI	1,456	1,035	958	77	5.42	10.85
Spambase	1,981	507	272	235	4.35	8.36

follows a similar trend, with 26, 33, and 52 regions for single, two, and three hidden layers. In contrast, the AGMP dataset shows moderate growth from 66 regions with a single layer to 116 with two layers, then escalating dramatically to 1,226 with three. The CDCDHI dataset also demonstrates rapid growth, increasing from 30 to 1,456 regions when moving from two to three hidden layers. The Spambase dataset exhibits more linear growth, showing 195 regions with a single layer, 357 with two layers, and 1,981 with three. Overall, simpler networks and datasets yield fewer regions, while more complex combinations result in significantly more regions, illustrating varying levels of decision boundary fragmentation.

4.2.2 Similarity Between Train and Test Regions

Gaining insights into the linear regions present during both training and testing is crucial for understanding a network's generalisation behaviour. Table 2 displays the discovery of regions during testing, indicating how many are similar or dissimilar to training, as well as the sample concentration patterns observed across all datasets and architectures. The dissimilar regions represent portions of the decision space that the network finds while using test samples, but are not available in training regions.

Region similarity: The similarity between training and testing regions indicates how well a model generalises to unseen data. Single hidden-layer networks often show strong similarity across most datasets. For instance, the Iris and CDCDHI datasets show perfect similarity, while the Seeds, AGMP, and Spambase datasets exhibit minimal dissimilarity. This indicates that shallow networks activate similar decision pathways for both training and testing samples, suggesting stable generalisation.

As the network depth increases, the complexity of the learned patterns also grows. In the Iris dataset with three hidden layers, 11 test regions were identified, of which 9 were similar to training regions, and 2 were not. The Seeds dataset contained 25 test regions, of which 17 were similar, and 8 were not. The AGMP dataset, also with three layers, produced 722 test regions: 636 similar to training and 86 exclusive to the test. The CDCDHI dataset generated 1035 test regions, 958 similar, and 77 not. Interestingly, the Spambase dataset showed a significant difference: it has 1981 training regions but only 507 test regions, resulting in 272 similarities and 235 differences. This suggests that even with fine-grained partitioning during training, test samples activate only a portion of those regions, revealing many new activation patterns.

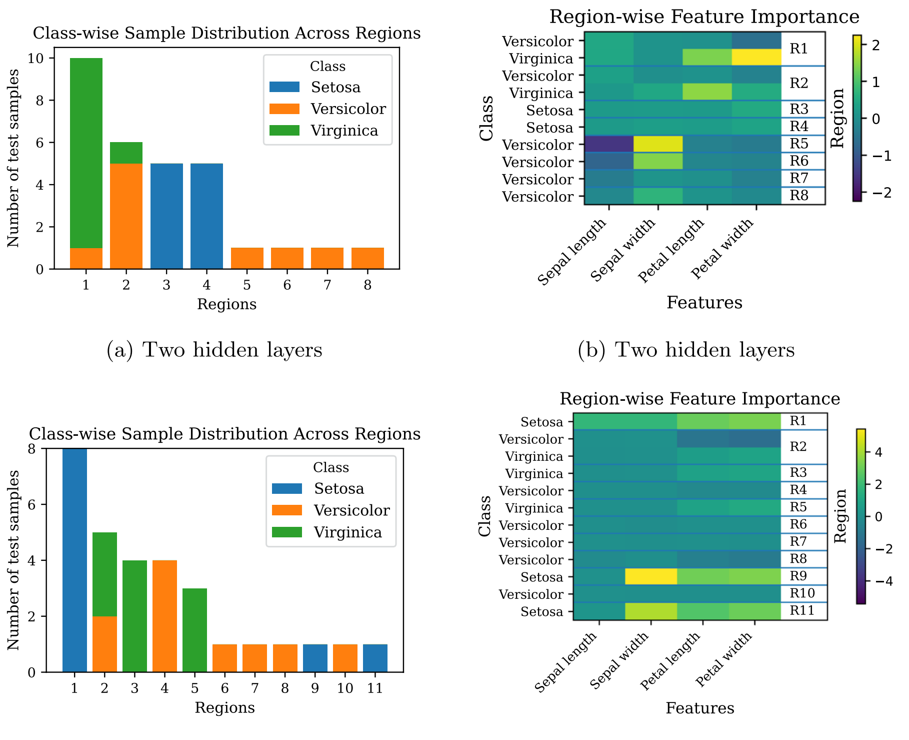
Sample coverage in regions: To evaluate test sample coverage across regions, two metrics were used. The *Top-1* measures the percentage of samples in the largest region, while *Top-C* combines larger regions based on *C*, which is the number of classes. For three classes scenarios such as Iris and Seeds, *C* is set to 3, whereas for binary classification tasks such as AGMP, CDCDHI, and Spambase, *C* is set to 2.

In shallow networks, sample grouping varies across datasets. For example, the Iris dataset shows strong grouping, with *Top-1* coverage of 46.67% and *Top-3* of 80%, indicating three main regions contain most of the test samples. The CDCDHI dataset shows even better grouping, with 66.61% for *Top-1* and 87.78% for *Top-2*, clustering around two main classes. In contrast, the Spambase dataset has a scattered distribution, with only 10.97% coverage for *Top-1* and 16.72% for *Top-2*, reflecting a wide spread of samples across many regions.

As networks deepen, coverage patterns shift. In the Iris dataset, *Top-1* coverage drops from 46.67% with a single layer to 26.67% with three layers, while *Top-3* coverage falls from 80.00 to 56.67%. The Seeds dataset follows a similar trend. For more complex datasets, fragmentation increases significantly. The AGMP dataset with three layers exhibits only 3.81% *Top-1* and 7.16% *Top-2* coverage, with 93% of samples spread across 720 regions. CDCDHI and Spambase display similar trends to AGMP. The results in Table 2 show that as the model increases in complexity, decision rules evolve from a few simple ones to many specialised local rules for handling diverse input patterns.

4.2.3 Region-Wise Sample Distribution and Feature Importance

The distribution of samples by region and the importance of features illustrate how samples are classified within distinct linear regions and which features primarily impact the model's decisions. Figure 2 presents this analysis for models with 2 and 3 hidden layers, while additional details for a single hidden layer are in Online Appendix C. The left side displays the distribution of test samples, and the right side shows averaged feature importance pro-



(a) Two hidden layers (b) Two hidden layers (c) Three hidden layers (d) Three hidden layers

Fig. 2 Sample distribution and feature importance across regions for the Iris dataset under two different network settings. Sub-figures **a**, **c** present sample distribution, and **b**, **d** represent feature importance

files. A region is considered pure for a class if all samples belong to it, with a 0.7 threshold applied when multiple classes are present, requiring at least 70% of samples to belong to the dominant class for labelling. This approach enhances decision-making while considering class overlap.

Figure 2a, b show clear class separation with the two largest regions demonstrating high purity. Region 1, mainly Virginica, has 90% purity and relies on petal length and width. Region 2, primarily Versicolor, has 83% purity and focuses on sepal length. Misclassifications occur when a false Virginica in Region 1 uses sepal features instead of petal features, and a false Versicolor in Region 2 has the same issue. Regions 3 and 4 are pure Setosa regions, balancing all features, while four separate Versicolor regions emphasise sepal width, with sepal length complicating classification and requiring reliance on other features for accurate categorisation.

Figure 2c, d show increased fragmentation in 11 regions, particularly in the confusion zone. Region 1 has 8 pure Setosa samples, while Region 2 has 5 samples with only 60% purity due to mixed contributions from Versicolor and Virginica. The heatmap indicates that pure Setosa regions, especially Region 1, rely heavily on all features. Singleton Setosa regions 9 and 10 focus on sepal and petal features. In Region 2, conflicting contributions confuse classification, with Versicolor showing negative petal values and Virginica showing positive ones. Virginica regions 3 and 5 depend mainly on petal features, while other pure Versicolor regions use different strategies with weak sepal contributions, highlighting unique decision pathways based on specific feature combinations.

4.2.4 Region Size and What-If Validity Analysis

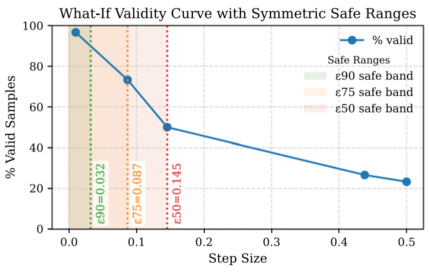
To enhance the sample distribution and feature importance analysis, region size analysis is performed, along with an evaluation of region stability through *what-if* perturbation tests. To measure the sizes of different regions and understand the limits of *what-if* analysis, the distance from each test sample to the nearest boundary of the ReLU was calculated. A neuron is at a decision boundary when its pre-activation value is zero. The normalised distance to this boundary for a neuron, denoted as ρ_j, is defined by the formula $\rho_j = |z_j^{(\ell)}|/\|\mathrm{w}_j^{(\ell)}\|_2$. The overall boundary distance for a sample x is determined as the smallest distance across all neurons in all hidden layers: $\rho(\mathrm{x}) = \min_{\ell,j} \rho_j$. This value indicates the smallest adjustment needed to change the sample into a different region.

For *what-if* analysis, the validity curve is used, which measures how many test samples stay in their original linear regions when faced with varying input perturbations. For each step size s, the percentage of samples with $\rho > s$ is determined, and these samples can withstand perturbations up to s without crossing region boundaries. Step sizes are chosen based on the empirical distribution of boundary distances, including the minimum distance, the first quartile (Q1), the median (Q2), the third quartile (Q3), and a conservative upper limit. The analysis reports three safe range thresholds: ε_{90} (10th percentile of ρ, ensuring 90% validity), ε_{75} (Q1, ensuring 75% validity), and ε_{50} (median, ensuring 50% validity). Figure 3 displays findings with *what-if* validity curves on the left and region size distributions on the right, using violin plots for 2 and 3 hidden layer settings with the iris dataset. The results for a single hidden layer show clear differences in the sizes of regions. More details are available in Online Appendix D.

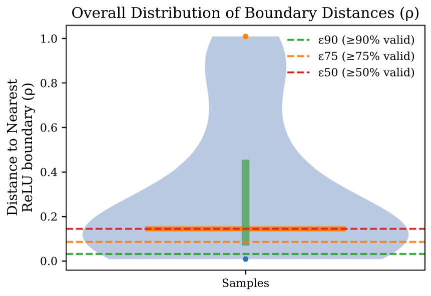
The two-layer network significantly reduces regions, with a median of 0.145. Figure 3b shows a violin plot illustrating this narrowing, resulting in an IQR of 0.351 and a range of 0.010 to 1.009. Safe thresholds have decreased: ε_{90} is 0.032, ε_{75} is 0.087, and ε_{50} is 0.145. The validity curve in Fig. 3a reveals an initial 96.7% validity at a step size of 0.01, dropping to 23.3% at 0.5, compared to 26.7% for the single-layer network. Overall, the reduced median and narrower IQR indicate that greater network depth leads to smaller, more uniform regions with less variability.

The reduction in region size and compression of distribution are evident in Fig. 3c, d for three layers. The median decreases to 0.109, the IQR narrows to 0.170, and the maximum value falls to 0.726. The violin plot in Fig. 3d shows concentrated lower values with minimal density above 0.4. Safe thresholds have been significantly lowered: $\varepsilon_{90} = 0.012$, $\varepsilon_{75} = 0.046$, and $\varepsilon_{50} = 0.109$. The validity curve in Fig. 3c shows a steep decline from 93.3 to 73.3% at ε_{75}, and down to 6.7% at a step size of 0.5, indicating network fragility. Perturbations over 0.5 units can invalidate 93% of samples in the deepest network compared to 73% in the shallowest.

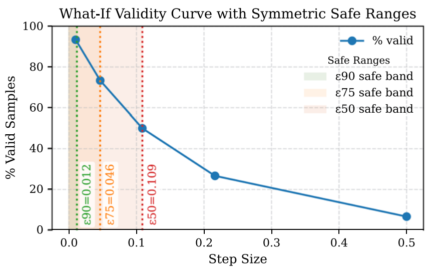
The results from Fig. 3 indicate that the distribution experiences asymmetric compression across different architectures. Although the validity curves converge at small perturbations, the patterns suggest that deeper networks tend to eliminate larger regions. This primarily affects stability under moderate to large perturbations rather than small ones.



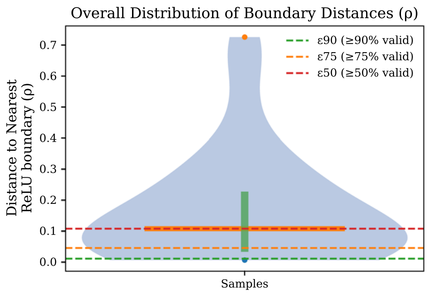
(a) Two hidden layers



(b) Two hidden layers



(c) Three hidden layers



(d) Three hidden layers

Fig. 3 Region size and *what-if* validity result for Iris dataset. Sub-figures **a** and **c** for validity curves and sub-figures **b** and **d** for violin plots of boundary distance distributions. The rows represent the following architectures: **a**, **b** for two hidden layers; **c**, **d** for three hidden layers

The analysis shows that deeper networks create more linear regions, affecting processing efficiency. Online Appendix E examines how L1 activation and L2 weight decay regularisation influence region formation and model stability. Results indicate the necessity of regularisation to control regions while maintaining classification performance.

4.3 Experimental Result on Decision Pathways

Decision boundaries across different architectures were analysed through region analysis. Sankey diagrams, created using sample number 5 from the Iris test set, illustrate how samples move through identified regions for both 2 and 3 hidden layer settings as presented in Fig. 4. Further information on the single hidden layer is available in Online Appendix F. Each network architecture recorded nonzero ReLU activations and calculated the absolute weight multiplied by the input for each connection, determining the thickness of links in the Sankey diagrams. This clearly shows each feature's contribution to the neurons and how their outputs influence the final class logit.

The Sankey diagram in Fig. 4a for two hidden layers identifies virginica and reveals the influence of petal width on neurons 2, 3, and 8. Petal length and sepal length strongly affect neurons 3 and 6, while sepal width impacts neurons 8, 3, 7, and 5 fairly evenly. Neurons 2, 3, and 8 are the main feature detectors in layer 1, with minor contributions from neurons 5, 6, and 7. In layer 2, neuron 1 integrates inputs from neurons 2, 3, and 8 and predominantly

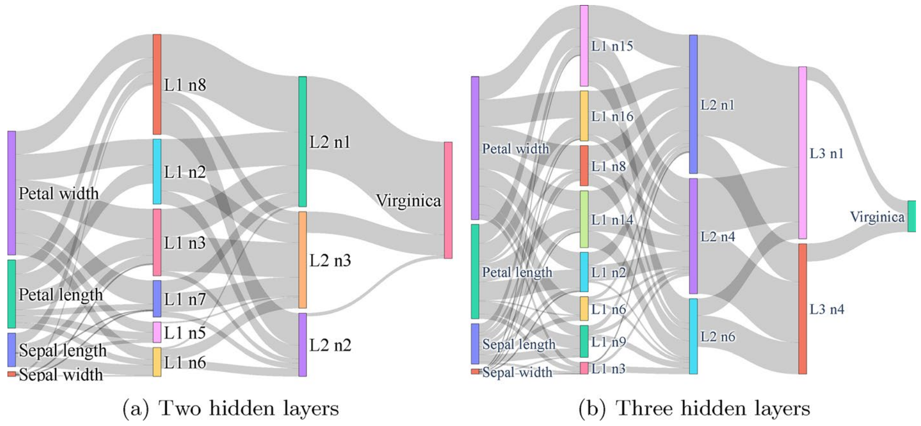


Fig. 4 Shanky diagram of two configurations using test sample number 5 from the Iris dataset: **a** with two hidden layers and **b** with three hidden layers

activates the virginica logit. This highlights how the network captures key features of petal width and length and channels them through neuron 1 for classification.

The three-hidden-layer model predicting virginica, shown in Fig. 4b , illustrates the flow of signals through 16 first-layer neurons, eight mid-level routers, and four top-level selectors. Petal length and width are the primary influences at the input, with petal width most strongly affecting neurons 6, 8, 12, 14, 15, and 16. Petal length impacts neurons 12, 14, 16, and 19, while sepal features show less influence. Neurons 14, 15, and 16 detect petal dimensions, and neurons 1 and 4 in layer 2 act as major routers, receiving strong signals from core first-layer neurons. The network efficiently consolidates information into three main mid-level streams before reaching the final layer.

4.4 Experimental Result on Feature Importance

By visualising how samples move through decision pathways in the network in different configurations, the next step is to quantify which input features influence these decisions at both local and global levels.

4.4.1 Experimental Result on Local Explanation (LIME, SHAP and Re3)

To quantify the influence of each input feature on the model's final confidence, per-feature probability contributions were calculated for each network depth. These contributions integrate the collapsed affine weights, input values, and the softmax Jacobian, clearly illustrating how each feature affects the predicted class probability. Per-feature probability contributions presented in Online Appendix G for a single sample across three different architecture settings, indicating which features significantly affect the model's confidence in its predictions. The values indicate that a deeper network relies more on the strongest signals.

To evaluate the contributions of features in the Re3 model, a comparison was conducted using LIME and SHAP. This analysis used a sample from the test set of the Iris dataset and examined models with different numbers of hidden layers. Although there were minor differences, the results for models with single and two hidden layers, detailed in Online

Appendix B, indicate that all three approaches agree that petal measurements are the primary factors influencing the network's decisions.

Figure 5 illustrates the results obtained from the model with three hidden layers. In Re3, depicted in Fig. 5a , petal width shows a strong positive influence on the prediction, while petal length also plays a significant role. Meanwhile, the contributions from sepal features largely cancel each other out. LIME, shown in Fig. 5b , exhibits a similar trend, but notes a slight negative effect for sepal width and a modest impact from sepal length. In SHAP, represented in Fig. 5c , petal width is again the most influential factor, followed by petal length, with sepal features nearly neutral in their contributions.

4.4.2 Experimental Result on Global Explanation (LIME, SHAP and Re3)

To assess how well the feature attributions of the proposed method align with established explainers at the dataset level, global importance scores were calculated by averaging the contributions of each feature across all Iris test samples for each architecture. Figure 6 compares the average scores for the proposed method, LIME, and SHAP across models with single, two, and three hidden layers. The subplots in Fig. 6 illustrate how each method ranks the sepal and petal dimensions when aggregated globally. This comparison provides a clear basis for evaluating their overall agreement and highlights any shifts in feature emphasis as the network depth increases.

Figure 6a shows the global feature importance for a single hidden layer model, where petal width is the most critical feature, followed by petal length, while sepal dimensions play a minor role. In Fig. 6b , the two-layer model indicates that petal width and length are nearly equal in importance for Re3, although LIME and SHAP still emphasise petal width more. By the three-layer model in Fig. 6c , the proposed method significantly amplifies the importance of petal width and also increases the significance of petal length, while sepal measurements drop to nearly zero. Despite minor differences, LIME and SHAP consistently

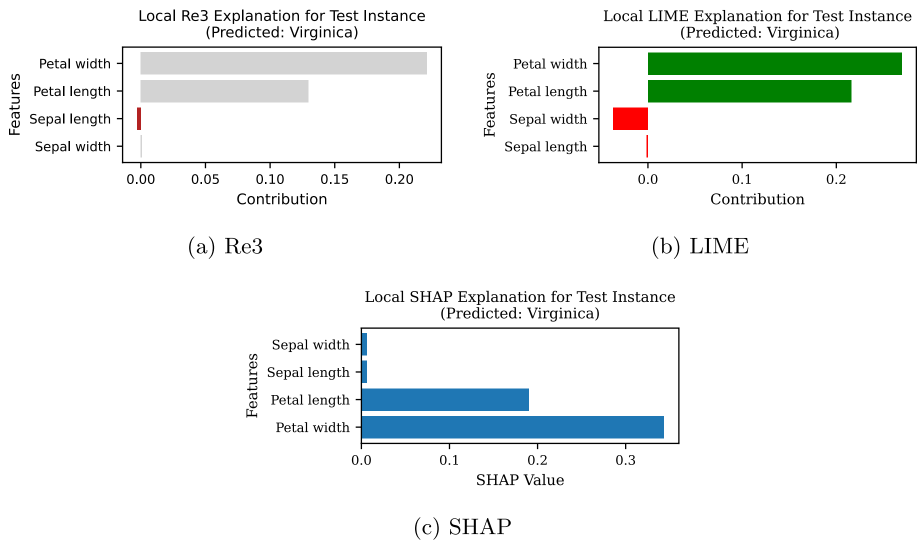


Fig. 5 Local interpretation of a sample from the Iris dataset using a model with three hidden layers, showing **a** Re3, **b** LIME, and **c** SHAP

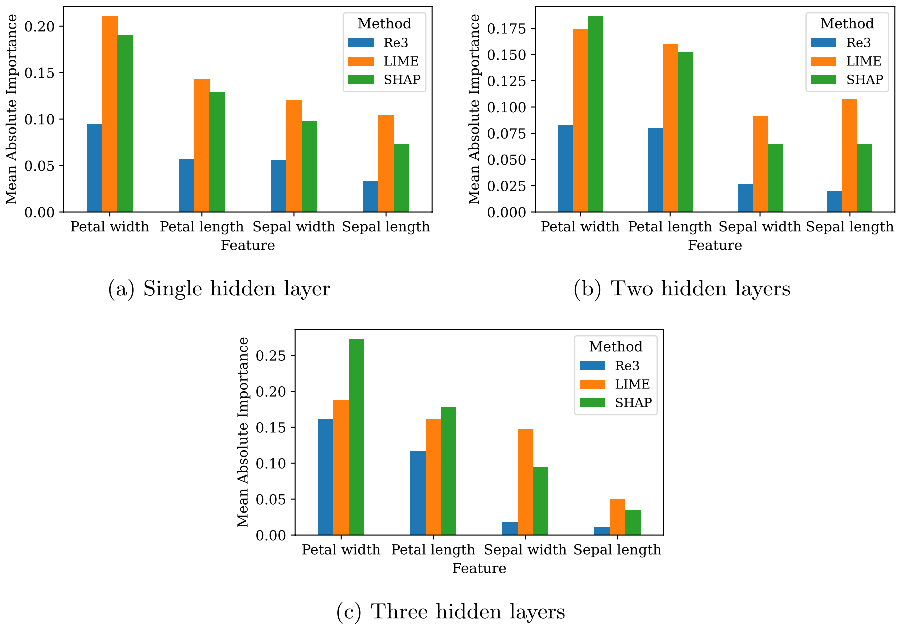


Fig. 6 Comparison of global explanations for the Iris dataset using LIME, SHAP, and Re3: **a** single hidden layer, **b** two hidden layers, and **c** three hidden layers

agree on the ranking, underscoring the importance of petal dimensions for Iris classification. These findings demonstrate that deeper architectures better highlight key features, with the proposed method aligning closely with established techniques.

4.4.3 Experimental Result on Statistical Evaluation

To measure how well the global feature rankings of the Re3 align with LIME and SHAP, three key metrics are used: Normalised Discounted Cumulative Gain (NDCG), Spearman's rank correlation, and Pearson's correlation. NDCG focuses on agreements at the top of rankings, Spearman's measures the correlation between ranking orders, and Pearson's assesses linear agreement of importance scores. Table 3 displays these metrics across datasets for models with single, two, and three hidden layers, highlighting agreements and significant differences among the methods.

In the Iris experiments, the proposed method aligns perfectly with LIME and SHAP in feature ordering, evidenced by NDCG and Spearman's ρ both reaching 1 at all depths. Additionally, Pearson's r exceeds 0.85, indicating nearly identical scores. For the Seeds dataset, NDCG remains above 0.80, with Spearman's ρ in the mid-0.90s for single and two layers, dropping to around 0.75 in the three-layer model. Pearson's r stays in the low-0.80s, reflecting minor changes in mid-ranked features.

In the AGMP dataset analysis, top rankings maintain strong NDCG values above 0.90, while Spearman's ρ ranges from 0.71 to 0.94 and Pearson's r varies from 0.68 to 0.90, indicating some shifts in less important sensor axes. The CDC diabetes indicators show high consistency, with NDCG values between 0.97 and 0.99, Spearman's ρ from 0.81 to

Table 3 Agreement between Re3 and LIME or SHAP evaluated using NDCG, Spearman, and Pearson correlations across three different NN depths and five datasets

Dataset	LIME vs Re3			SHAP vs Re3		
	NDCG	Spearman ρ	Pearson r	NDCG	Spearman	Pearson
Single hidden layer						
Iris	1.00	1.00	0.93	1.00	1.00	0.99
Seeds	0.99	0.92	0.98	0.98	0.75	0.89
AGMP	0.93	0.77	0.76	0.93	0.77	0.94
CDCDHI	0.97	0.90	0.89	0.99	0.97	0.99
Spambase	0.94	0.60	0.74	0.93	0.84	0.86
Two hidden layers						
Iris	1.00	0.80	0.87	1.00	1.00	0.97
Seeds	0.99	0.96	0.95	0.96	0.82	0.87
AGMP	0.93	0.88	0.82	0.99	0.94	0.97
CDCDHI	0.98	0.87	0.91	0.99	0.98	0.99
Spambase	0.94	0.61	0.74	0.95	0.85	0.85
Three hidden layers						
Iris	1.00	1.00	0.88	1.00	1.00	0.97
Seeds	0.92	0.92	0.74	0.86	0.75	0.69
AGMP	0.95	0.77	0.78	0.95	0.77	0.95
CDCDHI	0.95	0.84	0.84	0.99	0.92	0.98
Spambase	0.93	0.48	0.65	0.95	0.79	0.81

0.89, and Pearson's r ranging from 0.90 to 0.99, reflecting strong agreement among clinical features. Conversely, the Spambase dataset shows wider rank correlation variations, with Spearman's ρ from 0.47 to 0.62, despite a solid NDCG of 0.93–0.95 and Pearson's r from 0.69 to 0.84, highlighting differing rankings for less common words. Overall, these results confirm that the proposed method effectively replicates the top-ranking structure seen in LIME and SHAP across various datasets and models.

To evaluate the significance of the highest ranked features compared to the lowest, paired t-tests were conducted. This involved comparing the average importance of the top and bottom features. For the Iris dataset, the two most influential features were compared with the two least. For other datasets, the top five and bottom five features were analysed. Table 4 presents the average importance, differences, and results of the paired t-tests, including the t-value and p-value. A p-value below 0.05 is generally considered statistically significant.

In the single hidden layer models, the proposed approach shows a notable difference in feature importance between the top and bottom features of the Iris dataset, with a t value of 4.12 and a p value of 0.15, which is not significant. However, the differences in the larger Seeds, CDC, and Spambase datasets are significant, with the Spambase dataset showing a t value of 7.67 and a p value of 0.01. LIME and SHAP yield consistent results, highlighting significant differences in Seeds, CDC, and Spambase, but weaker evidence in Iris and AGMP.

Increasing the number of hidden layers improves data separation. The Iris model shows significance at $p < 0.05$, whereas other datasets exhibit very strong effects, such as the Spambase dataset, which shows $t = 10.18$ and $p < 0.001$. Both LIME and SHAP show clear differences between important and less important features across most datasets, indicating that deeper networks enhance these distinctions. The method with three hidden layers

Table 4 Paired t-tests to compare top and bottom feature importances across Re3, LIME, and SHAP methods for each dataset and model depth)

Dataset	Re3			LIME			SHAP		
	Diff	t	p	Diff	t	p	Diff	t	p
Single hidden layer									
Iris	0.030	4.124	0.151	0.064	2.510	0.241	0.074	4.512	0.138
Seeds	0.020	24.037	0.001	0.040	3.650	0.021	0.033	4.292	0.012
AGMP	0.002	1.890	0.131	0.003	1.543	0.197	0.003	1.993	0.117
CDC	0.026	4.874	0.008	0.038	7.508	0.002	0.033	5.412	0.005
Spambase	0.025	7.667	0.001	0.096	7.615	0.001	0.032	10.630	0.000
Two hidden layers									
Iris	0.058	35.962	0.017	0.067	68.034	0.009	0.105	7.842	0.080
Seeds	0.013	8.574	0.001	0.031	5.787	0.004	0.033	3.438	0.026
AGMP	0.002	2.362	0.077	0.003	1.640	0.176	0.002	1.124	0.100
CDC	0.022	6.085	0.003	0.039	7.970	0.001	0.031	5.661	0.004
Spambase	0.024	10.182	0.001	0.096	9.388	0.001	0.034	9.561	0.001
Three hidden layers									
Iris	0.102	9.199	0.068	0.076	2.182	0.273	0.159	10.134	0.062
Seeds	0.011	3.619	0.022	0.022	5.323	0.006	0.027	9.265	0.001
AGMP	0.003	2.432	0.071	0.004	1.219	0.289	0.004	1.932	0.125
CDC	0.025	5.953	0.004	0.039	7.847	0.001	0.032	5.359	0.005
Spambase	0.021	13.626	0.000	0.097	8.831	0.001	0.029	13.505	0.000

The reported values include the difference (Diff), t-statistic (t), and p value (p)

shows significant separations on datasets such as Seeds, CDC, and Spambase, whereas Iris and AGMP are near the 0.05 significance level. SHAP yields the highest t-statistics, such as $t = 13.63$ and $p < 0.001$ in Spambase, indicating a clear distinction between important and less important features. Overall, this approach, along with LIME and SHAP, effectively highlights crucial features, particularly in datasets with many inputs, with sharper differences in deeper networks.

4.4.4 Experimental Result on Stability Analysis

To evaluate the reliability of feature importance explanations, a stability analysis was conducted comparing Re3 with LIME and SHAP. This analysis was performed across 50 runs using different random seeds on 15 test samples with 3 hidden layers. Three metrics were measured: Spearman correlation of feature rankings, coefficient of variation of importance magnitudes, and top-1 feature changes. Table 5 summarises stability results across five datasets, where Re3 showed perfect stability, with a correlation of 1.000, a coefficient of variation of 0.000, and no changes in top-1 features. LIME proved to be much less stable, with correlations between 0.094 and 0.732, and many top-1 changes, ranging from 276 to 648 shifts. SHAP exhibited moderate stability, with correlations from 0.508 to 0.941, but its performance declined with high-dimensional data. The results suggest that sampling-based methods become less reliable as the number of features increases, while exact methods like Re3 provide consistent interpretations. A detailed explanation is provided in Online Appendix H.

Table 5 Stability comparison across 50 independent runs with different random seeds

Dataset	Method	Rank stability (Spearman ρ)	Mean CV	Top-1 changes
Iris	LIME	0.677 ± 0.109	0.071	302
	SHAP	0.941 ± 0.053	0.081	60
	Re3	1.000 ± 0.000	0.000	0
Seeds	LIME	0.671 ± 0.072	0.113	276
	SHAP	0.915 ± 0.064	0.086	84
	Re3	1.000 ± 0.000	0.000	0
AGMP	LIME	0.669 ± 0.028	0.176	233
	SHAP	0.807 ± 0.063	0.258	108
	Re3	1.000 ± 0.000	0.000	0
CDCDHI	LIME	0.732 ± 0.034	0.131	390
	SHAP	0.720 ± 0.055	0.294	160
	Re3	1.000 ± 0.000	0.000	0
Spambase	LIME	0.094 ± 0.022	0.185	648
	SHAP	0.508 ± 0.052	0.350	419
	Re3	1.000 ± 0.000	0.000	0

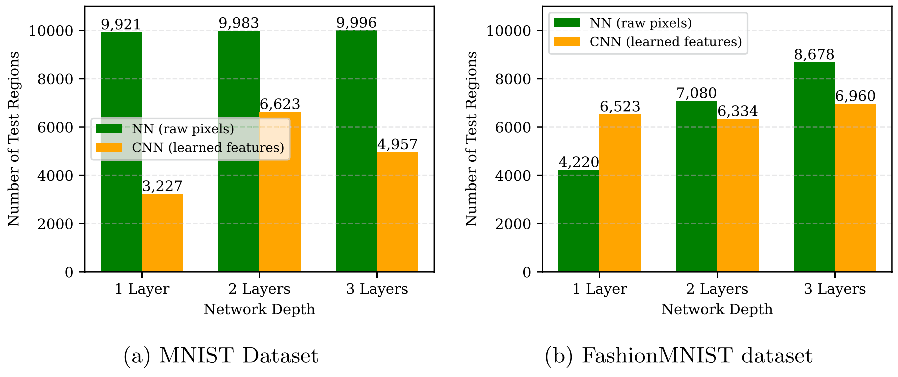


Fig. 7 Classwise sample distribution Iris

4.5 Experimental Result on Image Data

Extending Re3 from low-dimensional tabular data to high-dimensional image data poses computational and interpretability challenges. Raw pixel inputs can excessively fragment decision boundaries. This study explores the impact of convolutional feature extraction on region structure and Re3's scalability. Using the MNIST and Fashion-MNIST datasets, two architectural paradigms were compared: an NN that processes flattened 784-dimensional pixel vectors and a CNN architecture that extracts 128-dimensional features, which are then processed by MLP classifiers. Re3 was applied solely to the MLP, treating CNN layers as fixed preprocessing stages. Three depths were evaluated for both NN and CNN-MLP configurations to systematically analyse how feature learning affects decision structures.

The analysis in Fig. 7 reveals significant fragmentation in NNs. MNIST has a nearly one-to-one sample-to-region ratio, with 9921 to 9996 regions, indicating highly individualised decision boundaries. Fashion-MNIST has better clustering due to higher within-class variability, with 4220 to 8678 regions. CNN-MLP architectures reduce region counts for MNIST by 50 to 67%, ranging from 3227 to 6623 regions, and for Fashion-MNIST by 10 to 20%, between 6334 and 6960 regions. An exception occurs in a single-layer Fashion-MNIST scenario, where the number of regions increases from 4220 to 6523 due to shallow CNN fragmentation. The MNIST CNN-MLP exhibits a depth-dependent decrease in regions, from 6623 to 4957, suggesting that deeper networks refine features. In contrast, Fashion-MNIST maintains increased fragmentation, reflecting its complexity.

5 Discussion

In this section, every aspect of the proposed method and its key findings are analysed. The study began by posing four fundamental questions about NNs, as mentioned in the introduction (Sect. 1). These questions aim to uncover the inner mechanisms, neuron behaviour, and decision rules.

The use of established benchmark datasets ensures reproducibility and accessibility while requiring minimal preprocessing. This focus on ready-to-use datasets allows for a deeper examination of NN behaviour without extensive data wrangling. They provide consistent evaluation conditions, facilitating experiment replication. The selected datasets vary in dimensionality and sample size, maintaining sufficient complexity to clearly demonstrate neuron activation and feature attribution, while balancing clarity with effective illustration of the proposed algorithm's insights.

Simple NN architectures were chosen to make it easier to observe and understand how the network works. However, this doesn't mean the proposed algorithm is only for basic models. Using just one to three hidden layers with a small number of neurons keeps everything transparent. With only a few units, it's clear which neurons activate, how the two-layer setup creates a single affine map, and how each input feature influences the final output. Even in larger networks, the same calculations apply.

5.1 How Can the Internal Processes of NNs be Revealed and Explored?

NNs are complex due to their complicated mathematical relationships and numerous parameters, making it difficult to trace how inputs influence outputs. Understanding their functioning is crucial for identifying errors and ensuring trustworthy results.

LIME (Ribeiro et al., 2016) and SHAP (Lundberg & Lee, 2017) assess feature importance through input perturbation or cooperative game theory, but don't explain the internal workings of the network. Gradient-based methods like Integrated Gradients (Sundararajan et al., 2017) analyse model sensitivity but lack specific insights for each sample. In contrast, piecewise-affine analysis utilises the linear sections of ReLU-activated networks for precise, sample-specific mappings (Goodfellow et al., 2016; Montufar et al., 2014). Tools like OpenBox (Chu et al., 2018), and TropEx (Trimmel et al., 2021) identify affine regions but don't clarify the contributions of individual neurons and input features to predictions.

Re3 addresses this gap by calculating the exact affine map (A_r, D_r) for each input sample. It then analyses this map to show which neurons were activated (Online Appendix B), how many regions are there and decision-making process through regions (Sect. 4.2, how each raw feature passed through the network (Sect. 4.3) and as well as detailed attributions for each feature's logit and probability (Sect. 4.4.1). It also offers global ranking distributions (Sect. 4.4.2) and formal statistical tests (Sect. 4.4.3), all derived directly from the network's weights and masks, without using approximations or substitute models. Although other piecewise-linear activations, such as leaky ReLU or max-out, could theoretically achieve a similar result, standard ReLU is often preferred due to its straightforward zero threshold, inherent sparsity, and widespread use in real-world scenarios. Activations with smooth curves, such as sigmoid or tanh, do not have clear on/off boundaries, which makes it impossible to break them down into exact affine pieces without losing some information. Thus, ReLU-based piecewise analysis stands out as a uniquely effective method for uncovering the true workings of a network.

5.2 How Do NN's Create Decision Regions?

Discovery of regions (Sect. 4.2) reveals how ReLU networks systematically partition the input space, providing insights into decision structures. ReLU networks partition input space based on neuron activation, with each neuron's threshold separating active from inactive states (Montufar et al., 2014; Pascanu et al., 2013). Although larger networks can manage greater complexity, their performance is often constrained by data distribution and optimisation challenges.

The complexity of the data has a stronger influence on region counts than architectural parameters (Sect. 4.2.1), suggesting the need for adaptive partitioning strategies. Networks aim to improve how well they can represent different classes or categories. When the differences between these classes are clearer, the networks perform better. This is why adding more layers to a network helps only when the problem is more complex and needs more detailed divisions.

Class-coherent regions (Table 2) indicate that networks learn general decision patterns rather than merely memorising training examples. Simpler datasets show clear global structures, while complex datasets require specific rules due to intricate boundaries. Different regions show varying feature importance (Fig. 2), which challenges the idea that global explanation methods work the same everywhere. When regions highlight different feature combinations, aggregating data can obscure local decision-making. This suggests networks operate like groups of specialised classifiers instead of following a single global strategy. Adebayo et al. (2018) emphasise that understanding this is important to avoid misunderstanding from global summaries.

As depth increases, boundary distances shrink (Fig. 3), creating a trade-off: detailed distinctions can restrict the validity of counterfactual reasoning. This issue is common in local explanation methods (Lundberg & Lee, 2017; Ribeiro et al., 2016), which assume nearby points are similar. The practical use depends on requirements. (Wachter et al., 2017) suggest that useful counterfactuals involve minor changes with accurate local explanations. Defined limits help users assess the safety of changes, while significant alterations require global analysis methods. The depth-fragmentation relationship (Sect. 4.2.1) highlights the challenge of balancing the big picture with effective decision-making. Successful networks often divide tasks into smaller parts, making it difficult to see how they connect, even if each part is clear. This contrasts with the typical trade-off between being too broad or too specific; it's about understanding overall structure versus fine details.

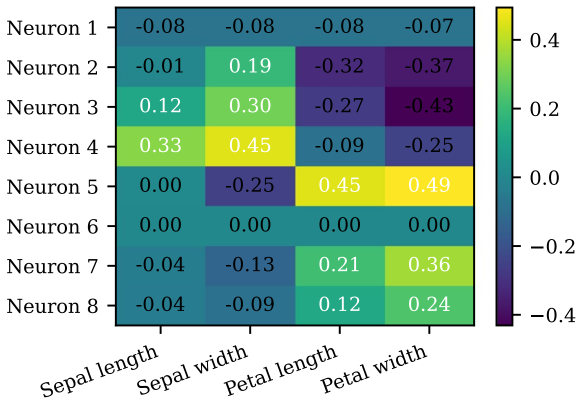
Analysing large networks with thousands of regions can be challenging. Sampling methods targeting high-activity areas offer a practical solution. Visualising complex shapes in high-dimensional spaces is simpler with summaries and statistics rather than direct displays. Additionally, architectural limitations restrict certain ReLU networks' use, affecting application in many modern systems, despite some still utilising them (Krizhevsky et al., 2012; Nair & Hinton, 2010). To sum up, analysing regions is crucial for organising networks, as these networks establish clear boundaries while accommodating diversity. Examining factors like the number of regions, their purity, and boundary distances uncovers complex decision-making patterns often missed with isolated examples. This approach helps evaluate decision complexity, differentiate specific rules from general ones, and validate explanations, shifting the focus from individual predictions to the overall structure of decision-making.

Standard regularisation techniques have inconsistent effects on the region count in networks (Online Appendix E). Shallow networks have minimal impact, whereas deeper networks yield variable results: some configurations reduce the number of regions, whereas others increase it. Notably, L2 weight decay can lead to more regions than unregularised models, suggesting that these penalties do not effectively target ReLU network structures. This limitation indicates that L1 and L2 regularisation focus on improving generalisation rather than controlling geometric partitioning. A better approach would involve a loss term that directly penalises diverse activation patterns to manage region proliferation and enhance scalability and interpretability in larger networks, marking an important area for future research.

5.3 How Do Neurons Get Activated and Interact Within the Hidden Layers?

Understanding neuron activity and identifying which neurons are more influential are crucial for interpreting and enhancing the network's performance (Lederer, 2021; Wang et al., 2021). The Re3 method aids in this analysis, but it's essential to verify its accuracy in reflecting the network's learning. The classification accuracy in Online Appendix A aligns perfectly with Re3 results, ensuring that insights on individual neurons genuinely represent the network's decision-making process, rather than misleading artefacts (Fig. 8).

Fig. 8 Final weights after completing all backpropagation



Analysing weight updates during backpropagation and final weight values provides limited insights into neuron contributions on a per-sample basis. To understand activation dynamics in the Re3 method, a binary mask is used, as explained in Eq. 10 (Sect. 3.3.1), to determine whether each hidden neuron is active for specific inputs. By aggregating these masks from test samples, it becomes possible to count the number of active versus inactive neurons (Online Appendix B) and rank the frequency of neurons among the top contributors, identifying the most important ones in the network.

These statistics highlight two important features of how networks function: sparsity and specialisation. Sparsity means that many neurons stay quiet and don't contribute to certain inputs, while specialisation shows that a small number of neurons often have the biggest impact on the output. For instance, in a single hidden-layer, Iris model with eight neurons, all of them might activate, but usually only one or two stand out in terms of how often they contribute. As more layers are added, this specialisation becomes even stronger, showing that deeper networks tend to cut away unnecessary neurons and focus the flow of information through a core group of influential ones. This natural sparsity helps prevent overfitting and makes it clear which parts of the model are actually doing the work. Overall, these patterns of activation and ranking provide a clear understanding of how the hidden units in a network collaborate to transform inputs into outputs.

5.4 How Do Features Move from the Input Through the Hidden Layers to the Final Decision of the Network?

The flow of information through the input, hidden, and output layers is vital for understanding NNs (Aggarwal et al., 2018). This process shows the abstraction of features at each layer, allowing researchers to analyse model behaviour, identify issues, and refine designs. The Re3 method's detailed per-sample calculations help determine each neuron's sensitivity to features and trace the information path through the network.

The use of a Sankey diagram (Fig. 4) clearly shows how transformations occur. In the single-layer network, feature weights are distributed across all active neurons, with the heaviest flows directed toward petal width and length. In the two-layer model, a few mid-level neurons combine signals related to petal and sepal traits before passing them on. By the third layer, a single top-layer neuron holds most of the decision weight for the virginica score.

The combined analysis of regions and Sankey charts shows how NNs operate. Early layers focus on gathering basic measurements, while the middle layers blend these into more complex size and shape channels. Finally, the last layers refine this information through a single pathway for making decisions. This approach reveals not only what each neuron is tracking but also how those signals move step by step to reach the final prediction. This dual perspective of *what* and *how* provides a clear, step-by-step view of how the network works inside.

5.5 Per-Feature Contribution and Ranking

One of the strengths of the proposed method is its ability to break down a single decision into the contributions of each feature (Online Appendix G). The Re3 approach calculates the exact probability contribution of each feature by simplifying the network into its per-sample affine form, (A_r, D_r), and then applying the softmax Jacobian to the logit contributions (Algorithm 3).

The method combines contributions from different features across the entire test set to create a comprehensive importance ranking (Sect. 4.4.2). This ranking shows which features consistently appear in top slots across samples, establishing a clear hierarchy. Petal width and petal length rank highly, indicating their significant role, while sepal length and width are less important (Fig. 6). These analyses provide a clear look at each individual prediction, as well as a broader understanding of which features are most important to the model. By showing both perspectives straight from the model's rules, Re3 creates a detailed map of how features influence predictions. This can help with debugging, selecting the right features, and simplifying the model.

5.6 Complementary Evaluation of LIME and SHAP

The main goal of this study was not to create a better explanation method than those already available, but to show that the insights gained from the new approach closely match results from well-known methods. Specifically, the probability contributions from the proposed approach follow similar patterns to those seen in LIME and SHAP (Online Appendix G, Fig. 5). The small differences in the exact numbers come from how Re3 uses direct transformations, compared to LIME's approach of perturbing data or SHAP's method of cooperative-game attributions. Despite this, the overall ranking and relative importance of the factors were consistent across all three methods.

When looking at the average contributions across the entire test set (Fig. 6), the agreement is even clearer. Regardless of using single, two, or three hidden layers, Re3, LIME, and SHAP consistently identify petal width as the most important feature, followed by petal length, with sepal width and length next. This consistent ranking indicates that Re3's analysis aligns with the findings from perturbation and gradient-based methods, suggesting it accurately captures key factors influencing the network's decisions.

Quantitative metrics provide strong support for the result. NDCG, Spearman's ρ, and Pearson's r values (Table 3) demonstrate that Re3 consistently identifies the most important signals, similar to LIME and SHAP, regardless of the model's complexity or depth. The paired t-tests comparing the top and bottom feature groups (Table 4) reveal that the most important inputs for Re3 show significantly higher average importance than the least important ones. This aligns with findings from LIME and SHAP. In most datasets, particularly those with many features, the distinction between these groups is highly significant and becomes more pronounced in deeper networks. These results confirm that Re3's attributions effectively differentiate between important and unimportant features.

The local case studies, global rankings, correlation metrics, and statistical tests show that Re3 offers important feature-importance results that enhance and support the findings from LIME and SHAP. By providing precise details at the neuron level and region level, Re3 deepens the understanding of how NNs make decisions while maintaining the agreement that practitioners depend on.

Table 6 positions Re3 within the wider context of NN interpretability methods, emphasising its distinct role as both an exact and computationally manageable approach for ReLU networks. The comparison shows that Re3 has a distinct role in interpretability. Unlike LIME and SHAP, which estimate feature importance through random sampling, Re3 pro-

Table 6 Comparison of Re3 with existing explainability methods for NNs

Method	Scope	Fidelity	Model requirements	Computational cost	Primary use case
LIME (Ribeiro et al., 2016)	Local	Approximate (stochastic)	Model-agnostic	$O(k \cdot d)$	Quick local approximations; interpretable linear surrogates
SHAP (Lundberg & Lee, 2017)	Local & Global	Approximate (stochastic) or Exact (trees)	Model-agnostic or tree-specific	$O(2^d)$ exact or $O(k \cdot d)$ sampling	Unified framework; global feature importance
OpenBox (Chu et al., 2018)	Global	Exact	Piecewise-linear	High (full enumeration)	Global analysis of all regions; small networks
TropEx (Trimmel et al., 2021)	Global	Exact	ReLU networks	High (polytope enum.)	Theoretical boundary analysis via tropical geometry
Re3	Local (sample and region)	Exact (deterministic)	ReLU networks	$O(L \cdot h \cdot d)$	Exact per-sample attribution; region-level analysis

vides exact and consistent explanations by using the piecewise-affine structure of ReLU networks. This method removes variability from sampling, ensuring that the same input always yields the same explanation, an important aspect for debugging and building trust. Re3 also avoids the high computational costs associated with global exact methods like OpenBox and TropEx, which require a comprehensive analysis of all linear regions or polytopes. By focusing on the specific region activated by each sample, Re3 achieves polynomial-time complexity per sample, making precise interpretability practical for moderately sized ReLU networks. This combination of accuracy and efficiency makes Re3 a valuable tool for those who need reliable and reproducible explanations without the added complexity of global analysis.

5.7 Contributions of the Study

The primary contribution of this work is to show that exact explanations for individual samples from ReLU networks can be achieved with a reasonable computational effort. By taking advantage of the known piecewise-linear nature of ReLU activations, Re3 extracts a specific linear mapping for each sample using just one forward pass and simple matrix operations. This makes the network's calculations easier to understand without needing to list all the linear regions, which previous exact methods required. The practical implementation (Algorithms 1–4) demonstrates that it is possible to achieve both exact results and manageable complexity for moderately sized ReLU networks.

The accessibility of Re3 goes beyond methodological rigour to encompass practical application. Its lightweight design, utilising standard libraries, facilitates easy integration into existing workflows with minimal additional coding requirements. By offering executable code in conjunction with the methodology, this research reduces barriers to adoption, enabling practitioners to validate and expand upon the approach. This emphasis on developer accessibility sets Re3 apart from methods that primarily exist in theoretical frameworks or necessitate specialised software.

Re3 connects local and global interpretability by providing explanations for individual samples and assessing feature importance across the entire dataset. The precise mathematical mappings enable the accurate attribution of specific predictions, while the aggregation of results across multiple samples reveals consistent trends in feature relevance (Fig. 6). This dual approach meets the distinct needs of interpretability, offering insights into specific decisions while also highlighting overarching patterns within a single framework, rather than relying on separate methods for local and global analyses.

Validation using LIME and SHAP demonstrates that the specific attributions provided by Re3 align closely with those of these well-established approximate methods while ensuring consistent and repeatable outcomes. The agreement observed across various datasets and models indicates that Re3 reliably identifies important features. Moreover, Re3 addresses a significant limitation associated with sampling-based methods explanation stability. Unlike LIME and SHAP, which may yield different explanations for the same input due to their inherent randomness, Re3 guarantees identical results for identical inputs. This reliability is essential for effective debugging and fostering trust in production systems.

Extending the Re3 framework to convolutional architectures underscores its relevance beyond fully connected networks. Experiments conducted on the MNIST and Fashion-MNIST datasets demonstrate that Re3 effectively processes features derived from CNN layers. However, the number of regions remains considerable, even when using convolutional preprocessing. This observation points to the fact that high-dimensional inputs inherently create complex decision boundaries. These results affirm Re3's practical application within ReLU networks, including CNNs, for moderately sized problems while recognising that its scalability to very large models is limited by the proliferation of regions.

The analysis of regions enhances understanding by showing how networks organise input data. By grouping samples with similar activation patterns, Re3 reveals important characteristics such as class purity, decision boundaries, and the consistency of feature importance across different groups. This approach addresses concerns raised by reviewers regarding the limited focus on single-sample explanations, demonstrating that the piecewise-affine framework enables analysis on various scales. Furthermore, characterising regions highlights areas of confusion where the model makes uncertain predictions, offering valuable insights for improving the model and assessing its reliability.

Re3's exact affine mappings enable deterministic *what-if* analysis within linear region boundaries. By algebraically manipulating feature values in the extracted mapping, practitioners can explore counterfactual scenarios without retraining or approximation. However, this capability is constrained by region boundary perturbations that alter activation patterns, requiring the mapping to be recomputed for the new region. The region size analysis provides practitioners with validity bounds, indicating the scope within which *what-if* explorations remain reliable. This represents a middle ground between unconstrained counterfactual generation and no counterfactual capability, offering exact predictions for perturbations within the current decision context.

6 Conclusion

Understanding NNs requires tools that can show how inputs affect outputs without sacrificing accuracy. This work introduces Re3, a method that derives precise per-sample affine mappings from trained ReLU networks by using their piecewise-linear structure. Through systematic evaluation on benchmark datasets, Re3 demonstrates that it is possible to achieve exact interpretability with a manageable computational cost, effectively bridging the gap between fast approximate methods and exhaustive exact approaches. The method offers deterministic feature attributions, stable global importance rankings, and regional analyses that illustrate how networks divide the input space. Validation using LIME and SHAP confirms alignment with established methods while enhancing reproducibility. Extending this analysis to convolutional architectures on the MNIST and Fashion-MNIST datasets shows its applicability beyond fully connected networks, although scalability issues arise with very large models. The region-level analysis uncovers decision structure patterns, class purity distributions, and consistent feature importance across sample groups, addressing interpretability at multiple scales within a unified framework.

Several areas deserve more focus. Creating custom regularisation techniques to manage region growth could make Re3 more scalable by reducing the number of linear partitions during training. Initial experiments with standard regularisation show limited results, suggesting that specific penalties designed for the piecewise-affine structure might lead to better results. Moreover, applying Re3 to CNN architectures and vision transformers could help clarify its design limits and needed changes. Finding ways to estimate uncertainty near region boundaries could improve what-if analyses by measuring the reliability of predictions for changes near activation transitions. Finally, using Re3 for specific issues in healthcare, finance, or autonomous systems could offer useful insights and guide necessary adjustments for safety-critical applications.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10994-025-06957-0.

Acknowledgements This study is part of projects (1) FitDrive, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 953432. (2) TRUSTY that has received funding from the SESAR 3 Joint Undertaking under grant agreement No 101114838 under the European Union's Horizon Europe research and innovation programme. 3) Trust_Gen_Z, funded by the VINNOVA, Diary No. 2024-01402. and 4) xBest, which is funded by the VR (Vetenskapsrådet - The Swedish Research Council), Diary No. 2024-05613.

Author Contributions Arnab Barua was responsible for the entire process, including conceptualisation, design, development, analysis, and manuscript writing. Mobyen Uddin Ahmed contributed to the study's design and planning and reformed the research questions. Shahina Begum assisted with explaining the results and discussion. Both Mobyen Uddin Ahmed and Shahina Begum reviewed the final version of the manuscript.

Funding Open access funding provided by Mälardalen University. This work was funded as mentioned in the Acknowledgements section.

Data Availability No datasets were generated or analysed during the current study.

Materials Availability Not applicable.

Code Availability The code used is available at Zonodo: https://doi.org/10.5281/zenodo.16535810.

Declarations

Competing interests The authors declare no competing interests.

Ethical Approval Not applicable.

Consent to Participate Not applicable.

Consent for Publication Not applicable.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency maps. In *Advances in neural information processing systems* (Vol. 31). ACM.

Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis, 22*(5), 717–727.

Aggarwal, C. C., et al. (2018). *Neural networks and deep learning* (Vol. 10). Springer.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE, 10*(7), 0130140.

Bak, S., Tran, H.-D., Hobbs, K., & Johnson, T. T. (2020). Improved geometric path enumeration for verifying relu neural networks. In *International conference on computer aided verification* (pp. 66–96). Springer.

Berzins, A. (2023). Polyhedral complex extraction from relu networks using edge subdivision. In *International conference on machine learning* (pp. 2234–2244). PMLR.

Christoph, M. (2020). *Interpretable machine learning: A guide for making black box models explainable*. SAGE.

Chu, L., Hu, X., Hu, J., Wang, L., & Pei, J. (2018). Exact and consistent interpretation for piecewise linear neural networks: A closed form solution. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1244–1253).

Covert, I., Lundberg, S., & Lee, S.-I. (2021). Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research, 22*(209), 1–90.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv preprint. arXiv:1702.08608

Garreau, D., & Luxburg, U. (2020). Explaining the explainer: A first theoretical analysis of lime. In *International conference on artificial intelligence and statistics* (pp. 1287–1296). PMLR.

Goodfellow, I., Bengio, Y., Courville, A.,& Bengio, Y. (2016). *Deep learning* (Vol. 1). MIT.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR), 51*(5), 1–42.

Hooker, S., Erhan, D., Kindermans, P.-J., & Kim, B. (2019). A benchmark for interpretability methods in deep neural networks. In *Advances in neural information processing systems* (Vol. 32).

Islam, M. R., Ahmed, M. U., Barua, S., & Begum, S. (2022). A systematic review of explainable artificial intelligence in terms of different application domains and tasks. *Applied Sciences, 12*(3), 1353.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (Vol. 25, pp. 1097–1105).

Kumar, I. E., Venkatasubramanian, S., Scheidegger, C., & Friedler, S. (2020). Problems with shapley-value-based explanations as feature importance measures. In *International conference on machine learning* (pp. 5491–5500). PMLR.

Lederer, J. (2021). Activation functions in artificial neural networks: A systematic overview. arXiv preprint. arXiv:2101.09957

Lipton, Z. C. (2018). The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue, 16*(3), 31–57.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision* (pp. 2736–2744).

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in neural information processing systems* (Vol. 30).

Madsen, A., Reddy, S., & Chandar, S. (2022). Post-hoc interpretability for neural NLP: A survey. *ACM Computing Surveys, 55*(8), 1–42.

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. arXiv preprint. arXiv:1611.06440

Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing, 73*, 1–15.

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

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences of the United States of America, 116*(44), 22071–22080.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning* (pp. 807–814).

Nanda, N., Chan, L., Lieberum, T., Smith, J., & Steinhardt, J. (2023). Progress measures for grokking via mechanistic interpretability. arXiv preprint. arXiv:2301.05217

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., & Mordvintsev, A. (2018). The building blocks of interpretability. *Distill, 3*(3), 10.

Pascanu, R., Montufar, G., & Bengio, Y. (2013). On the number of response regions of deep feed forward networks with piece-wise linear activations. arXiv preprint. arXiv:1312.6098

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

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). 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* (pp. 1135–1144).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533–536.

Salih, A. M., Raisi-Estabragh, Z., Galazzo, I. B., Radeva, P., Petersen, S. E., Lekadir, K., & Menegaz, G. (2025). A perspective on explainable artificial intelligence methods: Shap and lime. *Advanced Intelligent Systems, 7*(1), 2400304.

Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., & Müller, K.-R. (2021). Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE, 109*(3), 247–278.

Stadlhofer, A., & Mezhuyev, V. (2023). Approach to provide interpretability in machine learning models for image classification. *Industrial Artificial Intelligence, 1*(1), 10.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *International conference on machine learning* (pp. 3319–3328). PMLR.

Trimmel, M., Petzka, H., & Sminchisescu, C. (2021). Tropex: An algorithm for extracting linear terms in deep neural networks. In *International conference on learning representations*.

Velmurugan, M., Ouyang, C., Sindhgatta, R., & Moreira, C. (2023). Through the looking glass: Evaluating post hoc explanations using transparent models. *International Journal of Data Science and Analytics,* 20(2), 615–635.

Wachter, S., Mittelstadt, B., & Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology, 31*(2), 841–887.

Wang, L., Wang, C., Li, Y., & Wang, R. (2021). Explaining the behavior of neuron activations in deep neural networks. *Ad Hoc Networks, 111*, Article 102346.

Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V.I., Han, X., Gao, M., Lin, C.-Y., & Davis, L. S. (2018). NISP: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 9194–9203).