Diagnosis of Neural Network via Backward Deduction

Peifeng Yin, Lei Huang, Sunhwan Lee, Mu Qiao, Shubhi Asthana and Tagiga Nakamura
IBM Research - Almaden, San Jose, US
Email: peifengy@us.ibm.com lei.huang1@ibm.com {shlee, mqiao, sasthan, taiga}@us.ibm.com

***Abstract*—Although widely used in various areas, the Deep Neural Network suffers from the lack of interpretability. Existing works usually focus on one data instance and the found explanations are thus limited. We argue that a good understanding of a model should contain both systematic explanations of model behavior and effective detection of its vulnerability. Particularly we propose to use backward deduction to achieve these two goals. Given a constraint on the model output, the deduction backtracks the architecture to find corresponding data ranges in the input. In each layer, depending on the type, specific rules and/or algorithms are developed. The resulted ranges in the end can be interpreted by sampling exemplary data from them. In experiment we show that with different strategies in selecting the data ranges, the sampled fake data can either explain the model or reflect its vulnerability.**

***Index Terms*—deep neural network, backward deduction, diagnose, interpretation, vulnerability detection**

I. Introduction

Deep Neural Networks (DNN) has made significant progress in a lot of areas. However, they are usually treated as black boxes, suffering from the lack of interpretability. This problem may result in various issues such as losing user trust, missing fairness and even being vulnerable for adversarial attacks. For example, carefully-crafted input could purposely lead a model to misclassification [1], [2].

Existing works on explanation focus on one instance and aim to find its critical attribute(s) for the final classification [3]–[6]. These explanations are limited. We argue that a good understanding of a model should contain at least two systematic aspects. Firstly, it needs to show how the model is consistent with the training data. Secondly, it needs to report the model's vulnerability for adversarial attacks. We name this process as *Diagnosis*. To achieve these two goals, one should focus on the model itself.

The running of a Neural Network is a sequence of layer-based computation [7]. To diagnose the model, one intuition is to trace its flow in a backward order. That means, setting a constraint on a particular output, we aim to identify the corresponding input ranges layer by layer. Such backtrack runs in a strict compliance with logic and is similar to deductive reasoning (i.e., yielding true conclusions given the premises are true [8]). We thus refer to this process as *Deduction*.

In this work we propose a deduction framework to diagnose a neural network. It starts with an initial output constraint on the final layer, and deducts the corresponding input ranges. This process is conducted in an iterative way, where newly deducted input ranges of the current layer will serve as the next one's output constraints. For each layer type, we develop specific deductive rules and/or algorithms. The framework stops at the first layer with a set of deducted input data ranges. They can be interpreted by human user via sampled exemplary data instances. Many studies have shown that exemplar-based reasoning is fundamental to human understanding and cognitive perception [9]. Experiments demonstrate that with different ranking functions in selecting data ranges, those sampled data either show similar patterns with training data or expose model's vulnerability.

The key challenge lies in the deduction for affine projection layer. Geometrically, the task is to find a set of hyper rectangles to occupy an irregular high dimensional space. There are infinite solutions. We adopt a greedy algorithm, each time finding one hyper rectangle with maximal volume. The remaining spaces are then split into sub-problems to solve. This task is equivalent to select one maximal extreme point on the hyper plane to form a new hyper rectangle with the given minimal extreme point[1]. By replacing coordination basis, the problem is transformed to the one with an explicit and easily-computed solution. As the quantity of sub-spaces grows exponentially and there is no theoretic stop point, we also discuss options of ranking functions and stop criteria with respect to whether the training data is available or not.

Our contributions are in three folds. Firstly, we provided a unified framework to explain the model, and to detect its vulnerability using a deduction approach. Secondly, specific algorithms are proposed for backward deduction by introducing empirical metrics to efficiently check and tailor invalid spaces. Lastly, the results from experiments conducted with two synthetic data and one real world data provide a convincing evidence that the proposed approach can benefit the research community to pursue a different direction to gain better interpretability of a complex model like neural network.

The rest of the paper is organized as follows. Related work is introduced in section II. We formulate the deduction problem in Section III. Section IV presents detailed steps of our backward deduction framework to find the set of data ranges. Experimental results on two synthetic and one real datasets are shown in Section V, followed by the conclusion and future works in Section VI.

[1]Please refer to Section IV for detailed definitions

II. Related Work

Most existing methods for interpreting deep neural networks (DNN) are attribution methods, which can be generally grouped into two categories (1) perturbation-based methods, and (2) backpropagation-based methods.

To explain the predictions made by the deep learning model, perturbation-based methods firstly modify an input feature (or feature set) by removing, masking or altering them to generate a new input, which will be fed into the deep learning model to generate the corresponding new output. The attribution of the input feature can be directly computed by measuring the difference between the new output and original output [10]. The LIME [3] explains the predictions of various classifiers by learning an interpretable model locally around the prediction. Zhou and Troyanskaya [11] modified genomic sequence elements and measured those impact on the output. Perturbation-based methods can interpret model predictions to a certain extent, but they are usually computationally expensive as each perturbation requires a separate forward propagation through the network.

In contrast to perturbation-based methods, backpropagation-based methods only need one time forward and backward propagation through the network, but can compute attributions for all input features. The Gradient*Input [12] computes the partial output's derivatives in regard to each input feature, and assigns contribution score by measuring the difference between the activation of each neuron to its reference activation. The Integrated Gradients method [5] identifies two axioms *Sensitivity* and *Implementation Invariance* to guide the attribution method design, which computes the integrated gradient by cumulating gradients at all points that are along the straightline path. Moreover, Bach et al. [7] proposed a layer-wise relevance propatation (LRP) based method, which assumes that the classifier can be decomposed into several layers of computation. The LRP is computed with a backward pass, and the pixel contributions can be visualized by heatmap. The DeepLIFT [4] is also a backpropagation-based method, which decomposes the prediction of a neural network on a specific input by backpropagating the contributions of all neurons in the network to every input feature. There are many other methods such as saliency maps [13] showing how minor change to the input could affect the classification score based on the gradient information, PatternNet [14] providing neuron-wise explanations, and Deep-Taylor Decomposition [15] decomposing the classification decision as a sum of relevance scores.

Although various attribution methods have been proposed to help interpret DNNs, it is difficult to know how these methods are related and which one is better for a given DNN. Meanwhile, some attribution methods often generate inconsistent explanations [16]. Thus, Lundberg and Lee [17] proposed a unified framework SHAP based on existing explainability methods to improve the computational performance, and make explanations more robust, consistent, and align with human intuition. On the other hand, attribution methods may lack reliability when the explanation is sensitive to factors that do not contribute to the model prediction [18], and lack sensitivity to DNN parameter values, i.e., producing similar explanations for a DNN with randomly initialized weights and learned weights [19]. Many efforts have been done to interpret DNNs from other perspectives, such as self-explaining method [20] and bayesian non-parametric approach [21].

III. Problem Formulation

In this section we formulate the deduction problem as well as define basic concepts. Given a pre-trained neural network for classification, the diagnosis is to find what input would make the model classify this instance to be a particular class. In this work, we use the *data range* to describe the input characteristics.

Definition III.1 (Data Range)**.** *A data range* $r = \langle \mathbf{x}_{min}, \mathbf{x}_{max} \rangle$ *is represented by a pair of two vectors* $\mathbf{x}_{min}, \mathbf{x}_{max} \in \mathbb{R}^d$*, where* $\forall 1 \leq k \leq d, \mathbf{x}_{min}[k] \leq \mathbf{x}_{max}[k]$.

As can be seen in Definition III.1, a data range is a hyper rectangle in d-dimensional vector space. In the rest of the paper, if a data point $\mathbf{x} \in \mathbb{R}^d$ falls in a data range r, we denote it as $\mathbf{x} \in r$.

The diagnosis is to find a set of data ranges such that all data points falling in one of these ranges would be classified into a particular class. To be more general, we use a quantified probability range for the target class. The problem can be formulated as below:

Definition III.2 (Neural Network Diagnosis)**.** *Given a pre-trained neural network* $\mathtt{F}$ *for multi-label classification and a target label index* j*, the goal is to find a set of data ranges* $\mathbb{S} = \{r_1, \cdots, r_n\}$*, s.t.* $\forall \mathbf{x} \in \mathbb{R}^d$*, if* $\exists r \in \mathbb{S}$ *and* $\mathbf{x} \in r$*, then* $v_{min} \leq \mathtt{F}(\mathbf{x}) \leq 1$.

In practice, the lower-bound of the probability v_{min} depends on the desired confidence of the found data ranges, where larger value indicates higher confidence level of the model.

The prediction process of a neural network is essentially a concatenation of functions. Under Definition III.2, the diagnosis can be recursively done in a backward order, where the found data ranges in layer l are the output constraint for layer $l-1$. We define the layer diagnosis problem as below.

Definition III.3 (Layer Diagnosis)**.** *Given* l^{th} *layer function* $\mathtt{f}_l$ *and set of output constraints (data ranges)* $\mathbb{S}_{l+1}$*, i.e., diagnosis result at layer* $l+1$*, the goal is to find a set of input data ranges* $\mathbb{S}_l$*, s.t.* $\forall \mathbf{x} \in \mathbb{R}^{d_l}$*, if* $\exists r_l \in \mathbb{S}_l$ *and* $\mathbf{x} \in r_l$*, then* $\exists r_{l+1} \in \mathbb{S}_{l+1}$ *and* $\mathtt{f}_l(\mathbf{x}) \in r_{l+1}$.

We name the process of finding data ranges as *deduction*. For monotonic and element-wise functions such as sigmoid, hyperbolic and relu, the deduction is straightforward. For softmax layer, under Definition III.3, the problem can be converted to a particular case of affine projection layer. Therefore, we only focus on the dedution for affine projection in the rest of the work.

IV. Deduction for Affine Projection

The deduction for affine projection layer is to find a set of data ranges for the input so that the output falls in the desired ranges. The task can be split into several sub-tasks. Firstly, given a set of output data ranges, we can consider one at each time. In this case, the union of resulted input ranges for different outputs would be the final answer. Secondly, for multi-dimension output data ranges, we can focus on one dimension. Then the intersection of resulted input ranges would be the deducted explanation.

On the basis of Definition III.3, we define the sub-problem for affine projection with 1-dimension output below.

Definition IV.1 (1-dim Affine Deduction)**.** *For layer affine projection with one-dimension output, let* $\langle \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \rangle$ *denote the corresponding linear mapping vector and the scalar bias. Given the lower and upper bound* v_{min}, v_{max}*, the goal is to find a set of input data ranges* $\mathbb{S}$*, s.t.* $\forall \mathbf{x} \in \mathbb{R}^d$*, if* $\exists r \in \mathbb{S}$ *and* $\mathbf{x} \in r$*, the inequation always holds:* $v_{min} \leq \mathbf{w}^T \cdot \mathbf{x} + b \leq v_{max}$.

Note that the upper and lower bound inequation can be mutually transformed, i.e., $\mathbf{w}^T \cdot \mathbf{x} + b \leq v \leftrightarrow (-\mathbf{w})^T \cdot \mathbf{x} + (-b) \geq (-v)$. With no loss of generality, we assume the right half of the inequation ($\mathbf{w}^T \cdot \mathbf{x} + b \leq v$). But all discussed algorithms can be applied to the other half with a trivial transformation.

To solve the problem, we take three steps: i) generate a candidate data range, ii) check the validity of the candidate and iii) if not valid, tailor it to be a set of valid ranges. We discuss details of each one in the following subsections.

A. Generate Candidate Data Range

The deduction starts with generating a single data range, which provides an initial hyper rectangle to work on. Usually, this candidate should contain all possible data points.

Considering an affine projection layer, theoretically the input range should be an unbounded hyper rectangle. In practice, however, such input comes from either the last activation layer or the raw data feature. For the former case, the input would be bounded by the value range of the activation functions. For example, the hyperbolic activation results in the input to be $[-1, 1]$ for every dimension. For the latter case, the input would be bounded by the domain itself. For example, the gray picture leads to the input to be non-negative integers ranging from 0 to 255.

With an initial data range, we can do further pruning if the data that is used to train the model is also available. Depending on the purpose of deduction, this step could be different. We consider two cases. Firstly, the aim is to explain the model behavior consistent with the training data. Then the data range should contain as many data points as possible. With all training data, we can run the forward computation till the target layer, then filter those that are within the given bound. The candidate data range is the minimum hyper rectangle that contains all remaining data points.

Alternatively, the goal could be detecting the model vulnerability. That means, to find data ranges that are legal input but should not be classified to the class. In this case, the candidate should contain as less training data as possible. Following the same steps as above to obtain the minimum data range that contains all training data, we subtract the initial data range with this one. The resulted set of data ranges can be final candidates for following check and tailor.

B. Validity of Data Range

This step checks whether a candidate data range is valid to serve as the output range for last layer. Otherwise, whether it needs to be tailored or directly discarded.

Definition IV.2 (Valid Data Range)**.** *Given a one-dim affine projection* $\langle \mathbf{w}, b \rangle$*, a valid data range* r *with regarding to a upper bound* v_{max} *is the one whose all contained data points* $\forall \mathbf{x} \in r$ *satisfy such inequation* $\mathbf{w} \cdot \mathbf{x} + b \leq v_{max}$.

By definition, one needs to test all data points to check whether a data range is valid or not. This solution is definitely impractical as for continuous numbers, it is impossible to traverse all data points. Recall in Definition III.1, a data range is a hyper rectangle. Thus all points contained in a data range are bounded by its corner points. And we can only focus on the corner points for validity checking.

Definition IV.3 (Corner Point)**.** *Given a* d*-dimension data range* $r = \langle \mathbf{x}_{min}, \mathbf{x}_{max} \rangle$*, its corner point* $\mathbf{x}_c \in \mathbb{R}^d$ *is a data point in the same vector space and its element value is either from minimum or maximal bound, i.e.,* $\forall i \in \{1, 2, \cdots, d\}, \mathbf{x}_c[i] \in \{\mathbf{x}_{min}[i], \mathbf{x}_{max}[i]\}$.

Theorem 1. *Given an affine projection* $\langle \mathbf{w}, b \rangle$*, a data range* $r = \langle \mathbf{x}_{min}, \mathbf{x}_{max} \rangle$ *is valid if all of its corner points have a projected value no larger than the target upper bound* v_{max}.

Proof. By checking the sign of the $\mathbf{w}$ for each dimension, we construct such a corner point $\mathbf{x}_c$ that, $\forall i \in \{1, 2, \cdots, d\}$,

$$\mathbf{x}_c[i] = \begin{cases} \mathbf{x}_{max}[i] & \text{if } \texttt{sign}(\mathbf{w}[i]) > 0 \\ \mathbf{x}_{min}[i] & \text{if } \texttt{sign}(\mathbf{w}[i]) < 0 \\ 0 & \text{otherwise} \end{cases}$$

$\forall \mathbf{x} \in r$ and $\forall i \in \{1, 2, \cdots, d\}$, the inequation below is always true:

$$\texttt{sign}(\mathbf{w}[i])^T \cdot \mathbf{x}[i] \leq \texttt{sign}(\mathbf{w}[i])^T \cdot \mathbf{x}_c[i]$$

Thus we have $\mathbf{w}^T \cdot \mathbf{x} + b \leq \mathbf{w}^T \cdot \mathbf{x}_c + b \leq v_{max}$. □

The Theorem 1 provides a feasible way to check the validity of a candidate data range. However, it faces scalability issue as a d-dimension data range has 2^d corner points. The process of proof suggests there is a non-decreasing order among these corner points and we can only check the one with maximal projected value. Formally, we define such point as *maximal extreme point*.

Definition IV.4 (Maximal Extreme Point)**.** *Given a data range, its maximal extreme point* $\mathbf{c}_{max}$ *for an affine projection is such a corner point that achieves maximal projected value, i.e.,* $\mathbf{c}_{max} = \arg\max_{\mathbf{x} \in r} \mathbf{w}^T \cdot \mathbf{x} + b$.

Correspondingly, we can define the *minimal extreme point*:

Definition IV.5 (Minimal Extreme Point)**.** *Given a data range, its minimal extreme point $\mathbf{c}_{min}$ for an affine projection is such a corner point that achieves minimal projected value, i.e., $\mathbf{c}_{min} = \arg\min_{\mathbf{x} \in r} \mathbf{w}^T \cdot \mathbf{x} + b$.*

To check the validity of a data range, we first spend $O(d)$ to construct maximal and minimal extreme point $\mathbf{c}_{max}, \mathbf{c}_{min}$, then compare their projected values with target upper bound. There are in total three meaningful cases and we summarize them as well as corresponding actions in the following:

- The projected value of $\mathbf{c}_{max}$ is no larger than the upper bound. The whole data range is a valid one and can be directly used as output range for early layer deduction.
- The projected value of $\mathbf{c}_{min}$ is larger than the upper bound. This suggests the whole data range should be discarded.
- The projected value of $\mathbf{c}_{min}$ is no larger than the upper bound but not for $\mathbf{c}_{max}$. The data range is partially valid and needs to be tailored, which we discuss in Section IV-C.

C. Tailor Data Range Framework

In this step, the goal is to tailor a partially valid data range to a valid set of smaller data ranges. In general, after validity checking, these to-be-tailored data ranges have an invalid maximal extreme point and a valid minimal extreme point.

Geometrically, a partially valid data range is a hyper rectangle intercepted by a hyper plane. A 2-D example is illustrated in Figure 1a, where a rectangle is split by a straight line into two regions. The lower invalid part contains the maximal extreme point and upper valid one contains the minimal extreme point.

Tailoring a partially valid data range can be formulated as finding a new maximal extreme point to form a different data range with original minimal extreme point. We show three strategies from Figure 1b to 1d. The first one selects a point within invalid area, leading to another partially invalid data range. In Figure 1c, a new point is selected within the valid region. Although valid, the resulted range excludes too much valid spaces in original range. Therefore, the strategy of selecting new point exactly on the hyper plane, as shown in Figure 1d is the only acceptable one.

For a partially valid data range, it is impossible to cover the whole valid part with a single valid data range. When a maximal extreme point is selected on the plane, the formed new valid data range cuts the remaining part into several partially valid regions. Generally, there are d sub-ranges in a d-dimension space. Figure 2 illustrates the idea in 2-D space. To complete the tailoring step, we can recursively select new maximal extreme point for each sub-problem. Algorithm 1 shows the pseudo code of tailoring framework. The procedure works with given hyper plane weight $\mathbf{w}$, the upper bound v_{max} and the initial data range r_0. Note that we combine the affine projection's shift b with original upper bound for simplicity. In the while loop, a new maximal extreme point is selected for each sub-problem to form a valid data range. This step also generates d candidate data ranges by replacing the original $\mathbf{c}_{min}$'s value with found $\mathbf{c}_{max}$ for each dimension (line 12 to 14). Finally the algorithm stops with returning the set of valid data ranges $\mathbb{S}$.

Algorithm 1 Tailor Framework

```
1:  procedure FRAMEWORK(w, v_max, r_0)                          ▷ x ∈ r_0, w · x ≤ v_max
2:      S ← ∅, Q ← ∅                          ▷ initialize result set and Priority Queue
3:      Q.push(0, r_0)
4:      construct c_max_0 from r_0
5:      while Stopping criteria are not met ∧ Q ≠ ∅ do
6:          _, r ← Q.pop()
7:          construct c_min from r
8:          select c_max according to w, v_max, c_min, r
9:          construct valid data range r' from c_min, c_max
10:         S ← S ∪ {r'}
11:         for i ← 1, d do                    ▷ d-dimension vector space
12:             c_min_i ← c_min
13:             c_min_i[i] ← c_max[i]
14:             construct r_i from c_min_i, c_max_0
15:             score ← Rank(r_i)
16:             Q.push(score, r_i)
17:         end for
18:     end while
19:     return S
20: end procedure
```

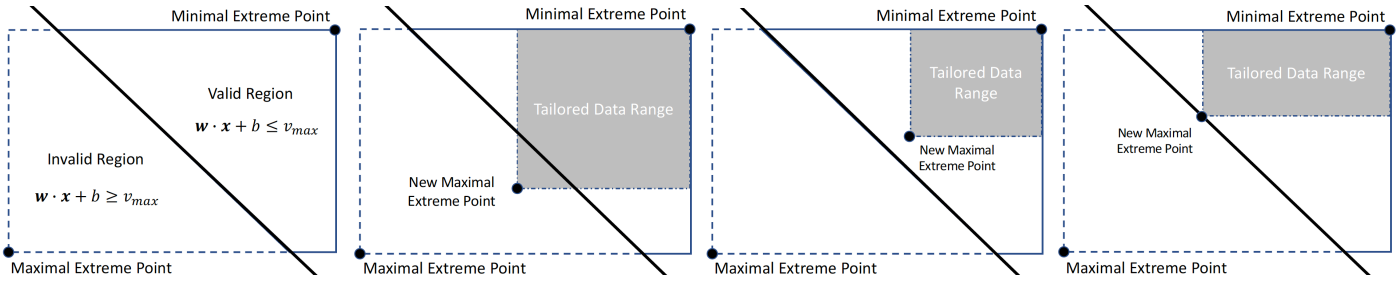
There are three issues that need to be addressed in this framework. Firstly, we need to provide a way to select the maximal extreme point (line 8). Also, as the quantity of the problems increases in a exponential rate, it is impractical to solve all of them. Therefore we need a mechanism to rank the sub-problems to determine which one to solve first (line 15). Finally, we need a way to decide when such recursion should stop (while loop in line 5). These issues are discussed one-by-one in the rest of the section.

D. Selection of Maximal Extreme Point

Even the choice space is narrowed down on the hyper plane, there are still infinite candidates. To further prune search space, we need to make use of *dominance* relation between two points.

Definition IV.6 (Dominance)**.** *Given an affine projection's weight $\mathbf{w}$, for two points $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^d$, if for every dimension $i \in \{1, 2, \cdots, d\}$, $\mathtt{sign}(\mathbf{w}[i])^T \cdot \mathbf{x}_1[i] \geq \mathtt{sign}(\mathbf{w}[i])^T \cdot \mathbf{x}_2[i]$, we define that point $\mathbf{x}_1$ is dominated by point $\mathbf{x}_2$ under $\mathbf{w}$.*

Combining the Definition of minimal extreme point in Definition IV.5 with the dominance, we know that all points that fall in the data range are dominated by the minimal extreme point. To select a new maximal extreme point, we consider several situations depending on whether the point is dominated by minimal extreme point and whether it is in the



(a) Original Data Range (b) Selection in Invalid Region (c) Selection in Valid Region (d) Selection on Hyper Plane

Fig. 1. Tailor Data Range by Finding New Maximal Extreme Point

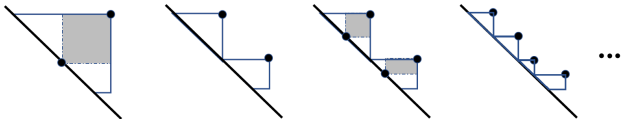


Fig. 2. Tailoring as an Infinitely Recursive Problem

original data range. Again to illustrate the concept, we show a 2-D example in Figure 3.

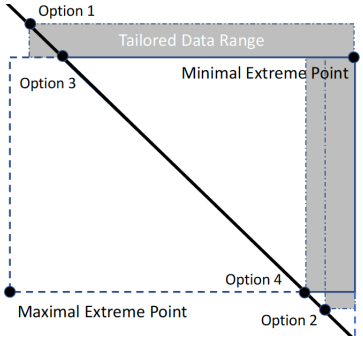


Fig. 3. New Maximal Extreme Point on Hyper Plane

The option 1 is neither within the data range nor dominated by the minimal extreme point. As can be seen, the resulted new range is partially valid and the valid part is not overlapped with the original one. The second candidate (option 2) is dominated by the minimal extreme point but falls out of the original data range. The obtained range is a valid one. However, there is a region that does not overlap with original one. Therefore, selection of candidate maximal extreme point should satisfy two conditions: i) on the hyper plane and ii) dominated by the minimal extreme point.

Particularly we consider two edge-cases, where the candidate point is on the edge of dominance or the data range. As shown in Figure 3, the "option 3" point is on the edge of dominance and it leads to an empty data range. Alternatively, if the point is on the other edge of data range (option 4), the obtained data range is a valid one, making the remaining original data range a new minimal extreme point. This scenario suggests different positions lead to different results and we need to consider which one would be optimal. In this work, we define the optimum one is the point where the resulted valid data range achieves maximal volume.

Definition IV.7 (Data Range Volume)**.** *Given a data range* $r = \{\mathbf{x}_{min}, \mathbf{x}_{max}\}$ *in* d*-dimensional space, its volume is defined as the product of length in each dimension, i.e.,* $\prod_{i=1}^{d} |\mathbf{x}_{max}[i] - \mathbf{x}_{min}[i]|$.

Definition IV.8 (Selection of Maximal Extreme Point)**.** *Given a hyper plane* $\mathbf{w}, v_{max}$ *and a partially valid data range* r *in* d*-dimension space, let* $\mathbf{c}_{min}$ *denote the minimal extreme point. The goal is to select a maximal extreme point on the intersection between hyper plane and the valid range so that the volume of resulted data range is maximized.*

$$\mathbf{c}_{max} = \arg\max_{\mathbf{x} \in r \wedge \mathbf{w}^T \cdot \mathbf{x} = v_{max}} \prod_{i=1}^{d} |\mathbf{x}[i] - \mathbf{c}_{min}[i]|$$

With no loss of generality, we assume that for each dimension i) the weight $\mathbf{w}$ is non-zero and ii) the data range's upper and lower bounds are not equal. Otherwise, the corresponding dimensions would either have no constraint or be constant. Then the problem would be reduced to a space with lower dimensions.

To solve the problem, we transform it to a simpler form via changing coordinate basis such that the minimal extreme point $\mathbf{c}_{min}$ is both the origin of the new coordinate and the lower bound of the data range r. Formally, let $\mathbf{e}_i$ denote the standard basis where the i^{th} element is 1 and others 0. The new basis is constructed with the sign of $\mathbf{c}_{min}$. We denote it as $d \times d$ matrix $\mathbf{A}$, where each column is a basis, represented as

$$\mathbf{A}_i = \texttt{sign}(\mathbf{c}_{max}[i] - \mathbf{c}_{min}[i])\mathbf{e}_i = \begin{cases} \mathbf{e}_i & \text{if } \mathbf{c}_{min}[i] < \mathbf{c}_{max}[i] \\ -\mathbf{e}_i & \text{otherwise} \end{cases} \tag{1}$$

It can be seen that the new basis is still orthogonal and normalized. Also, it is symmetric ($\mathbf{A}^T = \mathbf{A}$) and revertible. Particularly its inverse matrix is itself $\mathbf{A} = \mathbf{A}^{-1}$.

The transformation is shown as a two-step projection:

1) Shift the point by the vector $-\mathbf{c}_{min}$
2) Linearly map the point with matrix $\mathbf{A}$

Let $\mathbf{x} \in \mathbb{R}^d$ denote a data point under standard basis, after projection, the new data under new basis is $\mathbf{z} = \mathbf{A} \cdot (\mathbf{x} - \mathbf{c}_{min})$.

Conversely, given a data point $\mathbf{z}$ in new basis, the original point can be obtained by an inverse projection, i.e., $\mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{z} + \mathbf{c}_{min}$.

After the conversion, the data range consists of non-negative data points where the origin is its corner point. Formally, let $r_{\mathbf{A}} = \{\mathbf{0}, \mathbf{z}_{max}\}$ denote the data range, $\mathbf{w}_{\mathbf{A}}, v_{max_{\mathbf{A}}}$ the new hyper plane. The optimization problem is converted to:

$$\arg\max_{\mathbf{0}<\mathbf{z}\leq\mathbf{z}_{max} \wedge \mathbf{w}_{\mathbf{A}}^T \cdot \mathbf{z} = v_{max_{\mathbf{A}}}} \prod_{i=1}^{d} \mathbf{z}[i] \tag{2}$$

To solve the problem, we define the following objective function:

$$\mathcal{L}(\mathbf{z}) = \sum_{i=1}^{d} \log \mathbf{z}[i] + \lambda \cdot (v_{max_{\mathbf{A}}} - \mathbf{w}_{\mathbf{A}}^T \cdot \mathbf{z}) \tag{3}$$

where λ is a Lagrange multiplier. We can obtain the solution that maximize the objective function by setting the differentiation to be zero, i.e., $\frac{\mathtt{d}\mathcal{L}}{\mathtt{d}\mathbf{z}} = 0$. There exists an explicit solution $\mathbf{z}^*$:

$$\mathbf{z}^* = \frac{v_{max_{\mathbf{A}}}}{d} \cdot \frac{1}{\mathbf{w}_{\mathbf{A}}} \tag{4}$$

Note that the $\mathbf{w}_{\mathbf{A}}$ and $v_{max_{\mathbf{A}}}$ are all positive values, thus $\mathbf{z}^* > 0$. If for each dimension i we have $\mathbf{z}^*[i] \leq \mathbf{z}_{max}[i]$, then $\mathbf{z}^*$ is the final solution. Otherwise, for violated dimensions, we set it to be the upper bound of the data range and treat them as constant. The original problem is reduced to one with less variables. This iterative process continues until all dimensions of $\mathbf{z}$ are fixed. Finally we convert the solution to the space under originally standard basis, that is the optimal maximal extreme point as the projection itself keeps geometry properties such as distance and relative angle w.r.t $\mathbf{c}_{min}$.

Algorithm 2 Selection of New Maximal Extreme Point

```
1: procedure SELECTMAX(w, v_max, r)
2:     c_min, c_max ← construct min/max extreme points from r
3:     A ← construct new basis according to c_min, c_max
4:     w_A ← A^T · w
5:     v_max_A ← v_max − w^T · c_min
6:     z_max ← A · (c_max − c_min)
7:     z ← ∅^d, mask ← 1^d
8:     while sum(mask) > 0 do
9:         z[mask] ← (v_max_A / sum(mask)) · (1 / w_A[mask])
10:        if ∀1 ≤ i ≤ d, z[i] ≤ z_max[i] then
11:            break
12:        end if
13:        v_max_A ← v_max_A − w_A[z > z_max]^T · z_max[z > z_max]
14:        mask[z > z_max] ← 0          ▷ variables become constants
15:        z[z > z_max] ← z_max[z > z_max]
16:    end while
17:    return A^{-1} · z + c_min
18: end procedure
```

We show the pseudo code of selecting new maximal extreme point in Algorithm 2. It first constructs the new axis and converts the original problem to a simpler form (line 3 to 6). To solve the optimization problem, the algorithm starts with d variables (line 7). In the while loop, every time a solution is obtained according to Equation (4), it checks whether it is within the data range. If so, the loops is stopped. Otherwise, the violated dimension is set to be non-variable (line 14) and revised to be legally maximal value (line 15). Finally the algorithm converts the obtained complete solution back to original basis and returns it (line 17).

E. Ranking and Stopping Criteria

Here we discuss two remaining issues in Algorithm 1 tailoring framework, i.e., ranking and stopping criteria. As noted earlier, the tailoring problem is an infinitely recursive process with an exponential growth rate. Thus we need to solve problems with higher priority and stop the algorithm when some criteria is met. Similar to generation of candidate data range in Section IV-A, we have different strategies depending on whether the training is data available.

a) Training Data Unavailable: When the training data is not available, the ranking is to find the candidate that can potentially result in a large valid data range. Recall in Equation (4), the candidate minimal extreme point $\mathbf{c}_{min}$ only affects the value of $v_{max_{\mathbf{A}}}$, where a higher value leads to a larger volume. Given $v_{max_{\mathbf{A}}} = v_{max} - \mathbf{w}^T\mathbf{c}_{min}$, the $\mathbf{w}^T\mathbf{c}_{min}$ can serve as the ranking score, where smaller value indicates higher priority.

As for stopping criteria, there are two options, i.e., setting volume threshold or limiting the number of data ranges. Note that during the recursive process, there is an increasing order for the value of $\mathbf{w}^T\mathbf{c}_{min}$, suggesting the decreasing volume of candidate problems. The algorithm can stop if the top candidate problem has a value lower than a predefined threshold. Alternatively, we only return the top-k largest data range. In implementation, a candidate list of k valid data ranges are kept. Once the top candidate problems have a smaller estimated volume than the k^{th} data range, the algorithm can stop and return the current list.

b) Training Data Available: When training data is available, candidate partially valid data ranges can be ranked based on the number of data points dominated by their minimal extreme point $\mathbf{c}_{min}$. If the goal is to explain the model with training data patterns, then those with larger dominated instances are preferred. If the goal is to detect the vulnerability of the model, the one with lower dominated points is then preferred. Another variant would be the density, i.e., the number of dominated points divided by the volume.

It is similar for stopping criteria. Firstly, a dominance threshold can be predefined and the algorithm would stop if the top candidate has a number lower than the threshold. Alternatively, the algorithm can return top-k data ranges that contain most data points.

V. EVALUATION

In this section we conduct experiments to demonstrate the running of our deduction framework. Particularly we run three

experiments of which two are on synthetic data and one on real data. The first experiment is to diagnose a logistic regression on binary classification. It is on a 2-dimensional synthetic data that can be linearly separated. The second experiment is to diagnose a multi-layer neural network on multi-classification problem. To visualize the deduction result on each layer, we restrict the number of classes to be 2. But we do use the softmax layer as the decision output. Finally, we run experiments with MNIST data [22].

A. Diagnose Logistic Regression

The logistic regression can be treated as a special example of a multi-layer neural network without hidden layers. In this experiment we aim to demonstrate the running of our linear deduction as well as the impact of different initial probability boundaries on the final deduction results. For ease of visualization, we generate 1,000 2D synthetic data of two classes that can be linearly separated. And a logistic regression model is trained with 100% classification accuracy. The decision boundary for class 1 is 0.5. We change the initial output probability boundary (i.e., v_{min}) from 0.5 to 0.65 and visualize the deduction results in Figure 4.

As can be seen, the data of two classes distributed on two triangles respectively, with a clear boundary of straight line. The black rectangles represent the deducted data ranges for class 1 found by our deduction framework and they recursively occupy the space with given probability threshold. Furthermore, as threshold increases, these rectangles are moving away from the decision boundary, which is consistent with the model behavior.

B. Diagnose Neural Network

In this experiment, we train a multi-layer neural network on a data set that can not be linearly separated. Particularly, the neural network consists of two affine projection layers with a hyperbolic nonlinear activation in between, and it uses a softmax layer as the output normalization. For ease of visualization, we limit the hidden layer to 2 neurons. The model achieves 96.7% classification accuracy. We then run our framework to explain the decision for class 0 and plot in Figure 5 the data distribution and the deducted ranges in each layer in a backward order.

In Figure 5a, the framework finds two data ranges (represented as black rectangles) for which the softmax will output a probability bigger than 0.5 on the class 0. In hidden-to-output affine projection layer, these found data ranges serve as the output constraint and the framework then finds a series of input ranges that satisfy them, as shown in Figure 5b. One may note that some rectangles do not contain any data points. In next round of deduction, they may disappear as output constraints if the framework fail to find corresponding input data ranges. Figure 5c confirms it as all found data ranges contain some training data. Finally, Figure 5d shows the distribution of raw input data and the final explanations of the deduction. All black rectangles cover parts of the class 0 data (blue dots), all of which have an output probability no smaller than 0.5. Some data ranges contain areas with no data points. In practice, they are the potential vulnerability of model that can suffer from adversarial attacks.

C. Diagnose Neural Network on MNIST Data

In this experiment we explain the neural network aiming for classification on MNIST data set. Particularly, we pretrain a neural network with one hidden layer of 10 neurons. And we use hyperbolic as nonlinear activation function. Finally softmax is used as the output normalization. The model achieves 92.2% accuracy.

Since it is impossible to visualize resulted high-dimensional data ranges as in previous two experiments, we instead uniformly sample data points from each found data range and visualize them. During deduction, we adopt two separate ranking strategies, i.e., i) ranking on data coverage and ii) ranking on volume. The first one aims to find data range that is more consistent with training data while the second one aims to find vulnerability of the trained model.

Figure 6 shows a comparison of exemplary data with these two strategies. Note that all sampled data points have an output probability no smaller than 0.9 for the class. As can be seen in Figure 6a, strategy I results in clear sampled images that can be easily recognized. Also, data sampled from different ranges show different writing styles. This is reasonable as similar writing-style images are close to each other and thus more likely to fall in the same data range. In Figure 6b, sampled data points under strategy II ranking are quite vague and difficult to be recognized. This result suggests the potential vulnerability of the model for adversarial attack.

VI. Conclusion and Future Work

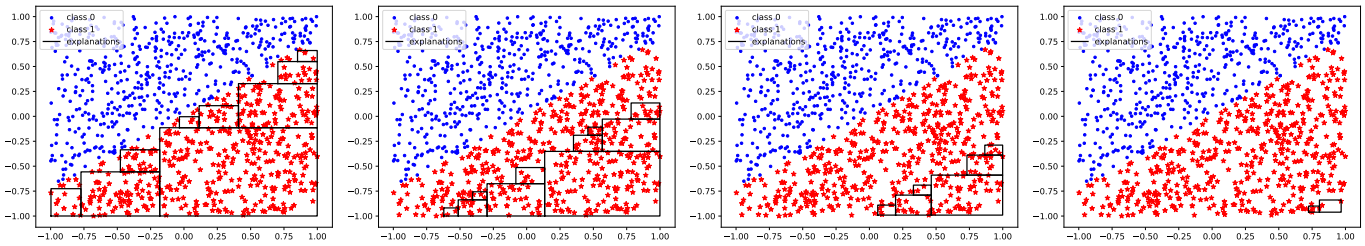
In this paper we discussed a novel method to explain a trained neural network as well as detecting its possible vulnerability by backward deduction, the process of finding data ranges per layer in a backward order. We formulated the problem as to find a series of hyper rectangles to cover the originally irregular space and developed a greedy algorithm to iteratively complete the task. To narrow the search space down, we leveraged the dominance relationship between points in a hyper rectangle, and defined ranking and stop criteria for both when training data is available and unavailable. Experiments on both synthetic and real data sets demonstrate promising deduction results.

For future work, we plan to extend the current framework for convolutional and recurrent neural network (CNN and RNN) layers. Also, as the iterative process of finding hyper rectangles is independent from each other, we plan to deploy the framework on a distributed and pipeline environment to improve the efficiency.

References

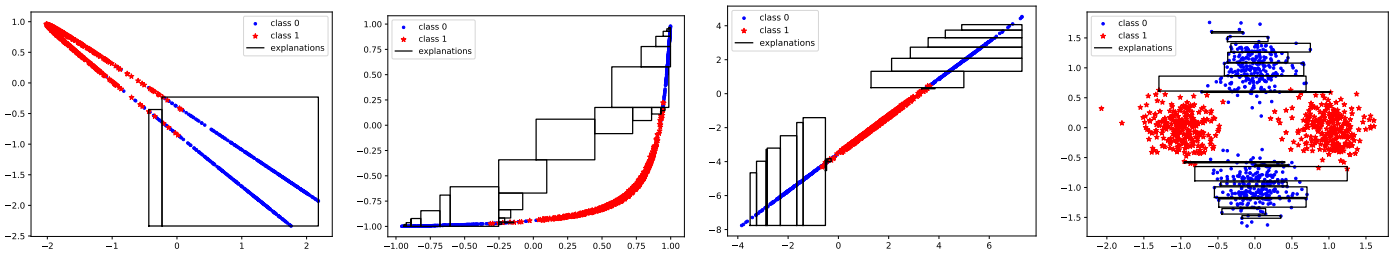
[1] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv 1412.6572*, 2014.

[2] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *IEEE European Symposium on Security and Privacy*, 2016, pp. 372–387.



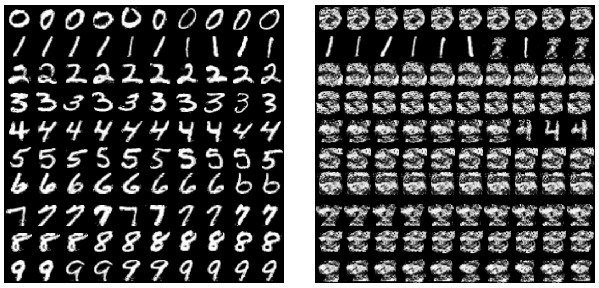
(a) 0.5 (b) 0.55 (c) 0.6 (d) 0.65

Fig. 4. Diagnosis of Logistic Regression for Class 1 with different probability thresholds



(a) Softmax (b) Affine II: Hidden to Output (c) Hyperbolic (d) Affine I: Input to Hidden

Fig. 5. Diagnosis of Neural Net for Class 0 in Each Layer



(a) Rank on Coverage (b) Rank on Volume

Fig. 6. Sampled Data from Deducted Data Range in MNIST

[3] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of ACM SIGKDD*, 2016, pp. 1135–1144.

[4] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," *arXiv 1704.02685*, 2017.

[5] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," *arXiv 1703.01365*, 2017.

[6] A. Dhurandhar, P. Chen, R. Luss, C. Tu, P. Ting, K. Shanmugam, and P. Das, "Explanations based on the missing: Towards contrastive explanations with pertinent negatives," in *Advances in Neural Information Processing Systems*, 2018, pp. 590–601.

[7] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Mller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLOS ONE*, vol. 10, pp. 1–46, 2015.

[8] P. N. Johnson-Laird, "Deductive reasoning," *Encyclopedia of Cognitive Science*, 2006.

[9] A. Newell and H. Simon, *Human problem solving*. Prentice-Hall Englewood Cliffs, 1972.

[10] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proceedings of ECCV*, 2014, pp. 818–833.

[11] J. Zhou and O. G Troyanskaya, "Predicting effects of noncoding variants with deep learning-based sequence model," *Nature methods*, vol. 12, 2015.

[12] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *arXiv 1605.01713*, 2016.

[13] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv 1312.6034*, 2013.

[14] P.-J. Kindermans, K. T. Schtt, M. Alber, K.-R. Mller, D. Erhan, B. Kim, and S. Dhne, "Learning how to explain neural networks: Patternnet and patternattribution," in *Proceedings of ICLR*, 2018.

[15] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Muller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211 – 222, 2017.

[16] S. M. Lundberg, G. G. Erion, and S. Lee, "Consistent individualized feature attribution for tree ensembles," *arXiv 1802.03888*, 2018.

[17] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, 2017, pp. 4765–4774.

[18] P.-J. Kindermans, S. Hooker, J. Adebayo, K. T. Schtt, M. Alber, S. Dhne, D. Erhan, and B. Kim, "The (un)reliability of saliency methods," *arXiv 1711.00867*, 2018.

[19] J. Adebayo, J. Gilmer, I. Goodfellow, and B. Kim, "Local explanation methods for deep neural networks lack sensitivity to parameter values," *arXiv 1810.03307*, 2018.

[20] D. Alvarez Melis and T. Jaakkola, "Towards robust interpretability with self-explaining neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7786–7795.

[21] W. Guo, S. Huang, Y. Tao, X. Xing, and L. Lin, "Explaining deep learning models – a bayesian non-parametric approach," in *Advances in Neural Information Processing Systems*, 2018, pp. 4519–4529.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.