界传播遇上约束简化：改进面向神经网络的基于逻辑的可解释人工智能

Ronaldo Gomes、Jairo Ribeiro、Luiz Queiroz、Thiago Alves Rocha*计算机科学系，塞阿拉联邦 ´学院（*IFCE*），马拉卡纳 ´乌，巴西{ronaldo.apoliano.gomes05、jairo.duarte.ribeiro01、luiz.fernando.paulino60}@aluno.ifce.edu.br，*thiago.alves@ifce.edu.br ✉

arXiv:2603.01923v1 [cs.LO] 2 Mar 2026

摘要——用于解释神经网络决策的基于逻辑的方法对正确性与非冗余性提供形式化保证，但它们往往计算成本很高，尤其是对于大型网络。在本工作中，我们通过将界传播与约束简化相结合来提高此类方法的效率。这些由传播导出的简化收紧了神经元界限，并消除了不必要的二元变量，从而使解释过程更加高效。我们的实验表明，结合这些技术可将解释时间最多减少89.26%，对较大的神经网络尤为明显。

索引词—神经网络、可解释人工智能、基于逻辑的可解释性、区间分析。

I. 引言

神经网络（NN）在处理高度复杂的分类任务方面展现出巨大潜力 [1]。尽管取得了成功，它们通常仍被视为不透明算法 [2]。这意味着，对于给定的一组输入，其产生输出的过程本质上并不可解释或阐明。在此背景下，对于给定的输入及其对应的输出，解释被定义为一组直接促成最终输出的属性。换言之，这些属性足以确定结果，而其他属性则并非必要。例如，给定输入 $\{sneeze = True,\ weight = 70\ kg,\ headache = True,\ age = 40\ years\}$ 及其神经网络输出流感，一种可能的解释是 $\{sneeze = True, headache = True\}$。也就是说，如果一个实例具有 $sneeze = True$ 和 $headache = True$，那么神经网络的预测就是流感，无论 $weight$ 和 age 的值如何。

启发式可解释人工智能（可解释人工智能）方法，例如 LIME [3]和 Anchors [4]，，已被用于为神经网络及其他机器学习（ML）模型提供解释。然而，这些方法缺乏正确性保证。当不存在任何实例使得其属性值符合解释中的指定值、但神经网络却给出不同预测时，才能提供正确性保证。近期已开发出基于逻辑的可解释人工智能方法，为多种机器学习模型提供具有此类保证的解释 [5, 6, 7, 8, 9, 10, 11, 12]。虽然一些近期的可解释性方法也提供形式化保证，但它们通常关注鲁棒性或稳定性，例如确保微小的输入扰动不会改变解释 [13, 14]。尽管这些保证很重要，但它们并不能确保解释在正确性意义上足以被证明能够复现模型的输出。

*作者感谢 CNPq 和 CAPES 提供部分经费支持。

相比之下，Ignatiev 等人 [6] 提出的这类基于逻辑的方法，能够提供既无冗余又可证明正确的解释；这正是本研究关注的保证概念。该方法使用逻辑约束——由涉及二元变量和连续变量的线性（不）等式的布尔组合构成——来计算无冗余且可证明正确的神经网络解释。随后，使用混合整数线性规划（MILP）验证输入属性的某个子集是否足以复现模型的输出。尽管该方法确保了解释的正确性和简洁性，但将其应用于大型神经网络时会面临可扩展性方面的挑战。

在本研究中，我们提出一种方法，通过利用神经元的过近似界传播，改进神经网络的逻辑型可解释人工智能。在某些情况下，该方法无需使用 MILP 求解器，因为我们可以判定某个输入属性对于解释并非必要。过近似方法是可靠但不完备的：如果该方法判定某个输入属性并非必要，则这一结论必定正确；然而，如果它判定该属性是必要的，该属性实际上仍有可能并非必需。该方法之所以可靠，是因为它所生成的界必定包含实际值；但它并不完备，因为这些界可能比实际值的范围更宽。因此，在无法判定某个输入属性对于解释并非必要的情况下，我们会利用传播得到的界来简化逻辑约束。随后，我们使用简化后的约束和 MILP 求解器，判断能否将该输入属性从解释中排除。通过这种方法，我们旨在开发一种能够确保正确性、为神经网络提供无冗余解释，同时提升可扩展性的方案。

在实验中，我们比较了我们的方法与 Ignatiev 等人提出的原始方法的解释时间 [6]。结果显示，我们的方法取得了显著改进，在规模较大的神经网络中尤其如此。例如，在一个案例中，我们的方法将解释时间缩短了 89.00% 以上。这些发现表明，将简洁的界传播方法与约束简化相结合，有望提升基于逻辑的可解释人工智能的可扩展性，使解释生成更加高效，同时不影响正确性。

在本研究中，我们采用了一种简单且粗略的界传播方法，称为 Box。现有关于神经网络验证的文献 [15, 16, 17] 已对 Box 及其他界

传播技术在鲁棒性分析中的应用进行了广泛研究。尽管更先进的方法（如 Zonotope [16] 和 DeepPoly [17],）能够提供更精确的界估计，但计算成本也更高。探索这些更精确的界传播方法将留待未来工作。

在发挥近似界传播优势的基础上，近期的神经网络验证研究 [18, 19] 也致力于将这些技术与 MILP 求解器或直接分支定界方法相结合，以增强鲁棒性认证。通常，这些方法会交替使用松弛技术和 MILP 求解器，在验证过程的不同阶段逐步收紧界。相比之下，我们的可解释人工智能方法从始至终持续应用界传播。如果传播无法明确判定某个输入属性对于解释是否不必要，我们便利用传播过程中获得的信息，在求助于 MILP 求解器之前简化约束。在其他情况下，仅靠传播就足以得出结论，因此完全无需使用 MILP 求解器。

II. 背景

本节旨在介绍理解本文所需的基本概念。

A. 线性实数算术上的一阶逻辑

在本文中，我们使用一阶逻辑（FOL）给出具有正确性保证的解释。我们使用线性实数算术理论（LRA）上的无量词一阶公式。因此，一阶变量可以取实数值 $\mathbb{R}$。详情请参见 [20]。据此，我们将公式定义如下：

$$\begin{aligned} F, G &:= p \mid (F \wedge G) \mid (F \vee G) \mid (\neg F) \mid (F \to G), \\ p &:= \sum_{i=1}^{n} w_i x_i \leq b, \end{aligned} \tag{1}$$

使得 F 和 G 是在线性实数算术理论上的无量词一阶公式。此外，p表示满足 $n \geq 1$的原子公式，其中每个 w_i 和 b 都是固定的实数，而每个 x_i 都是一阶变量。注意，我们允许使用其他字母来表示变量，而不局限于 x_i，例如 s_i、z_i、q_i。例如，按此定义，$(2.5x_1 + 3.1x_2 \geq 6) \wedge (x_1 = 1 \vee x_1 = 2) \wedge (x_1 = 2 \to x_2 \leq 1.1)$ 是一个公式。注意，我们允许使用标准缩写，例如将$\neg(2.5x_1 + 3.1x_2 < 6)$作为 $2.5x_1 + 3.1x_2 \geq 6$的缩写。

由于我们假定公式的语义建立在实数域之上，一个赋值 $\mathcal{A}$（对应于公式 F）是从 F 的一阶变量到实数域中元素的映射。例如，$\{x_1 \mapsto 2.3, x_2 \mapsto 1\}$ 是$(2.5x_1 + 3.1x_2 \geq 6) \wedge (x_1 = 1 \vee x_1 = 2) \wedge (x_1 = 2 \to x_2 \leq 1.1)$的一个赋值。一个赋值 $\mathcal{A}$ 满足公式F，如果 F 在该赋值下为真。例如，$\{x_1 \mapsto 2,\ x_2 \mapsto 1.05\}$ 满足上述示例中的公式，而 $\{x_1 \mapsto 2.3, x_2 \mapsto 1\}$ 并不满足它。此外，一个赋值 $\mathcal{A}$满足公式集合 Γ，如果 Γ 中的所有公式在 $\mathcal{A}$下都为真。

公式集 Γ是可满足的，如果存在 Γ的满足赋值。例如，集合 $\{(2.5x_1 + 3.1x_2 \geq 6),\ (x_1 = 1 \vee x_1 = 2),\ (x_1 = 2 \to x_2 \leq 1.1)\}$ 是可满足的，因为 $\{x_1 \mapsto 2, x_2 \mapsto 1.05\}$ 满足它。再举一例，集合 $\{(x_1 \geq 2), (x_1 < 1)\}$ 是不可满足的，因为没有任何赋值满足它。给定公式集 Γ 和公式G，记号 $\Gamma \models G$ 用于表示逻辑后承或蕴涵，即每个满足 Γ 的赋值也满足 G。作为说明性示例，设 $\Gamma = \{x_1 = 2, x_2 \geq 1\}$和 $G = (2.5x_1 + x_2 \geq 5) \wedge (x_1 = 1 \vee x_1 = 2)$。则 $\Gamma \models G$。蕴涵的本质在于基于前提集 Γ确保结论 G 的正确性。在计算解释的语境中，如 [6], 所述，逻辑后承是保证神经网络所作预测之正确性的基本工具。

可满足性与蕴涵之间的关系是逻辑的一个基本方面。众所周知，对于所有公式集合 Γ 和所有公式 G，有 $\Gamma \models G$当且仅当 $\Gamma \cup \{\neg G\}$ 不可满足。例如，$\{x_1 = 2, x_2 \geq 1) \neg((2.5x_1 + x_2 \geq 5) \wedge (x_1 = 1 \vee x_1 = 2))\}$ 没有满足赋值，因为满足 $\{x_1 = 2, x_2 \geq 1\}$的赋值也满足$(2.5x_1 + x_2 \geq 5) \wedge (x_1 = 1 \vee x_1 = 2)$，因此不满足 $\neg((2.5x_1 + x_2 \geq 5) \wedge (x_1 = 1 \vee x_1 = 2))$。由于我们的方法建立在逻辑后承的概念之上，因此可以在为神经网络计算解释的语境中利用这一联系。

B. 分类问题与神经网络

在机器学习中，分类问题定义在一组 n 输入属性 $\mathcal{F} = \{x_1, ..., x_n\}$ 和一组 k 类别 $\mathcal{K} = \{c_1, c_2, ..., c_k\}$之上。在本文中，我们假设每个输入属性 $x_i \in \mathcal{F}$ 的取值 v_i 都来自实数域。此外，每个输入属性x_i 都有一个上界 ub_i 和一个下界 lb_i，满足$lb_i \leq x_i \leq ub_i$，其定义域为闭区间 $[lb_i, ub_i]$。这表示为一组域约束，或输入空间 $\mathcal{D} = \{lb_1 \leq x_1 \leq ub_1,\ lb_2 \leq x_2 \leq ub_2, ..., lb_n \leq x_n \leq ub_n\}$。例如，表示一个人身高的属性属于实数域，其下界和上界分别可以是0.5米和2.1米。此外，$\mathcal{I} = \{x_1 = v_1, x_2 = v_2, ..., x_n = v_n\}$ 表示输入空间中的一个特定点或实例，其中每个 v_i 都属于 x_i的定义域。

神经网络是一个将输入空间中的元素映射到类别集合 $\mathcal{K}$ 的函数。前馈神经网络由 $L + 1$ 层神经元组成。每一层 $l \in \{0, 1, ..., L\}$ 由 n_l 个神经元组成，编号从 1 到 n_l。这些层和神经元共同定义了神经网络的架构。第 0 层是虚构的，对应神经网络的输入；最后一层 L 对应其输出。第 1 层至 $L - 1$ 层通常称为隐藏层。设 x_i^l 为第 i层的第 l个神经元的输出，其中 $i \in \{1, ..., n_l\}$。神经网络的输入可表示为 x_i^0，或简写为 x_i。此外，输出可表示为 x_i^L，或简写为 o_i。

给定层 l 中的神经元值 x_i^l 由前一层的输出值 x_j^{l-1} 计算得出，其中

$j \in \{1, ..., n_{l-1}\}$。每个神经元都会对前一层中各神经元的输出进行线性组合。随后，该神经元会应用一个非线性函数，也称为激活函数。线性部分的输出表示为 $\sum_{j=1}^{n_{l-1}} w_{i,\ j}^{l} x_j^{l-1} + b_i^l$ 其中 $w_{i,\ j}^{l}$ 和 b_i^l 分别表示权重和偏置，它们是第 i 个层 l神经元的参数。本文仅考虑以修正线性单元（ReLU）作为激活函数的前馈神经网络，因为 ReLU 具有分段线性特性，可以用线性约束表示。ReLU 是一种广泛使用的激活函数，其输出是输入值与零之间的较大值。因此，$x_i^l = \text{ReLU}(\sum_{j=1}^{n_{l-1}} w_{i,\ j}^{l} x_j^{l-1} + b_i^l)$ 即为 ReLU 的输出。最后一层 L 由 $n_L = k$ 个神经元组成，每个类别对应一个。通常还会使用 Softmax层对输出层进行归一化。因此，这些值表示各类别对应的概率。概率最高的类别将被选为预测类别。然而，无需考虑这种归一化变换，因为它不会改变最后一层的最大值。因此，预测类别为满足 $i = \arg\max_{j \in \{1,...,k\}} x_j^L$ 的 $c_i \in \mathcal{K}$。因此，我们将神经网络表示为函数 $\mathcal{M}$，使得对于所有实例 $\mathcal{I}$，$\mathcal{M}(\mathcal{I}) = c \in \mathcal{K}$。

C. MILP – 混合整数线性规划

在混合整数线性规划（MILP）中，目标是在满足线性约束的条件下优化线性函数，其中部分或全部变量必须为整数 [21]。MILP 是我们工作中的一项关键技术，用于确定神经网络中每个神经元的下界与上界。例如，我们利用最小化问题来确定神经网络中神经元的下界。此过程涉及构建一个目标函数，以求得下界的最小值，同时满足反映神经网络行为的约束。为说明 MILP 的结构，我们在下面给出一个示例：

$$\begin{aligned} \min \quad & y_1 \\ \text{s.t.} \quad & 1 \le x_1 \le 3 \\ & 3x_1 - 2 \le y_1 \\ & y_1 \le 3x_1 - 2 - 0.5(1 - z_1) \\ & 0 \le y_1 \le 8z_1 \\ & z_1 \in \{0, 1\} \end{aligned} \tag{2}$$

在式（2）的 MILP 中，我们要为变量 x_1、y_1、z_1 找出取值，使满足约束的所有取值中目标函数 y_1的值最小。变量 z_1 是二元变量，因为 $z_1 \in \{0, 1\}$ 是 MILP 中的一个约束，而变量 x_1、y_1 的定义域为实数 $\mathbb{R}$。

一个重要的观察是，正如第 II-A 小节所述，不含目标函数的 MILP 问题对应于一个可满足性问题。鉴于计算解释的方法依赖于逻辑后承，并考虑到可满足性与逻辑后承之间的联系，我们采用 MILP 求解器来处理解释任务。此外，在构建混合整数线性规划模型的过程中，我们可以利用优化，具体而言，使用 MILP 求解器来确定神经网络中神经元的紧下界与上界。这些紧界在训练完成后、生成解释之前立即计算。在解释阶段，我们仅依赖过近似界传播，从而避免实时求解混合整数线性规划问题所带来的计算开销。

III. 面向神经网络的基于逻辑的可解释人工智能

如前所述，大多数用于解释机器学习模型的方法都采用启发式，因此其解释无法得到完全信任。基于逻辑的可解释性能够提供正确且无冗余的结果。对于神经网络，Ignatiev 等人 [6] 采用了一种基于逻辑的方法，该方法基于包含二元变量和连续变量的线性约束。给定一个输入，该方法会找出一个输入属性子集，足以证明神经网络给出的相应输出合理。该方法保证了解释的正确性和最小性，这类解释称为溯因解释。溯因解释是构成一条规则的属性子集。应用该规则时，可保证模型给出相同的预测。下面的形式化定义概括了这一概念。

定义 1（溯因解释）：设 $\mathcal{I} = \{x_1 = v_1, ..., x_n = v_n\}$ 为一个输入实例，$\mathcal{M}$ 为一个神经网络，且 $\mathcal{M}(\mathcal{I}) = c \in \mathcal{K}$。若对于所有 $v_1' \in [l_1, u_1], ..., v_n' \in [l_n, u_n]$，当 $v_j' = v_j \quad x_j = v_j \in \mathcal{X} \quad \mathcal{M}\{x_1 = v_1', ..., x_n =$对每个 均 $v_n'\} = c$ 成立时，则 ( ) 。溯因解释 $\mathcal{X}$是 $\mathcal{I}$的一个最小子集。

最小性确保解释 $\mathcal{X}$ 不包含任何冗余属性。换言之，从 $\mathcal{X}$ 中移除任何属性 $x_j = v_j$，都会得到一个子集，而该子集无法再保证在定义的界内得到相同的预测 c。需要注意的是，给定实例可能有多个不同的溯因解释。不同的特征子集可能分别满足解释条件，同时保持最小性。这意味着，可以通过多种方式为某个预测提供依据，每种方式都突显了实例中足以确保相同输出的不同方面。

Ignatiev 等人 [6] 提出的获取溯因解释的方法如下。首先，将神经网络和输入空间 $\mathcal{D} = \{lb_1 \le x_1 \le ub_1,\ lb_2 \le x_2 \le ub_2, ..., lb_n \le x_n \le ub_n\}$ 编码为一组公式 $\mathcal{F}$。此外，给定输入空间中的一个实例 $\mathcal{I} = \{x_1 = v_1, x_2 = v_2, ..., x_n = v_n\}$、神经网络 $\mathcal{M}$ 及其对应的预测$\mathcal{M}(\mathcal{I}) = c_j \in \mathcal{K}$，我们将这一事实编码为公式 E。该公式断言，o_j 是所有输出神经元中最大的：

$$E = \bigwedge_{i=1, i \neq j}^{k} o_j > o_i.$$

然后，$\mathcal{I} \cup \mathcal{F}$是可满足的，并且 $\mathcal{I} \cup \mathcal{F} \models E$。溯因解释 $\mathcal{X}$ 通过从$\mathcal{I}$中逐个移除属性来计算。例如，给定 $x_i = v_i \in \mathcal{I}$，若 $(\mathcal{I} \backslash \{x_i = v_i\}) \cup \mathcal{F} \models E$，则属性 x_i 在解释中并非必要，因而被移除。否则，若 $(\mathcal{I} \setminus \{x_i = v_i\}) \cup \mathcal{F} \not\models E$，则保留 x_i，因为无法保证同一类别。该过程被描述

在算法 1 中，并对所有属性执行。然后， $\mathcal{X}$就是该过程结束时的结果。这意味着，对于 $\mathcal{X}$ 中的属性值，无论其余属性取何值，神经网络都会作出相同的分类。

算法 1计算解释 $\mathcal{X}$

输入： $\mathcal{F}$， $\mathcal{I}$， E

$\mathcal{X} \leftarrow \mathcal{I}$

对于 $x_i = v_i \in \mathcal{I}$，**执行**

  如果 $(\mathcal{X} \setminus \{x_i = v_i\}) \cup \mathcal{F} \models E$，**则**

    $\mathcal{X} \leftarrow \mathcal{X} \setminus \{x_i = v_i\}$

返回 $\mathcal{X}$

由于检验蕴涵关系 $(\mathcal{X} \setminus \{x_i = v_i\}) \cup \mathcal{F} \models E$等价于检验 $(\mathcal{X} \setminus \{x_i = v_i\}) \cup \mathcal{F} \cup \{\neg E\}$是否不可满足，且 $\mathcal{F}$、$\mathcal{X}$ 和 $\neg E$ 被编码为带有连续变量和二元变量的线性约束，因此此类蕴涵可由混合整数线性规划求解器处理。在本工作中，我们采用Tjeng 等人 [22] 提出的编码，将给定的神经网络表示为一组公式 $\mathcal{F}$。公式（4）至（9）描述了该编码，其中 $l = 1, \ldots, L-1$，并且对于每个 l， $j = 1, \ldots, n_l$。

$$lb_i \leq x_i \leq ub_i, \quad i = 1, \ldots, n \tag{3}$$

$$x_j^l \leq \sum_{i=1}^{n_{l-1}} w_{j,i}^l x_i^{l-1} + b_j^l - lb_j^l (1 - z_j^l) \tag{4}$$

$$x_j^l \geq \sum_{i=1}^{n_{l-1}} w_{j,i}^l x_i^{l-1} + b_j^l \tag{5}$$

$$x_j^l \leq ub_j^l z_j^l \tag{6}$$

$$z_j^l \in \{0, 1\} \tag{7}$$

$$x_j^l \geq 0 \tag{8}$$

$$o_j = \sum_{i=1}^{n_{L-1}} w_{j,i}^L x_i^{L-1} + b_j^L, \quad j = 1, \ldots, n_L \tag{9}$$

在公式（4）至（9）中，常数 ub_j^l 表示第 l层第 j个神经元的上界。类似地，常数 lb_j^l 表示下界。界 ub_j^l 通过将变量x_j^l 与后续层中的其他约束隔离来定义。然后，对 x_j^l 进行最大化以求得其上界。采用类似过程可求得下界 lb_j^l。可以使用 MILP 求解器执行此过程，以获得更精确的界。由于输入属性的界已知，因此可以进行此项优化。此外，这些界还可以帮助求解器加速解释的计算。

在需要生成解释之前，可以使用 MILP 求解器并考虑输入属性的整个定义域，从而找到这些紧界。在针对给定实例的解释过程中，我们按照本文提出的方法使用宽松界。由于在寻找解释的过程中，某些输入属性会被固定（即 $x_i = v_i$），如果过近似界传播得出了更紧的界，我们就用解释过程中得到的界替换原始界。这些更紧的界可以帮助混合整数线性规划求解器

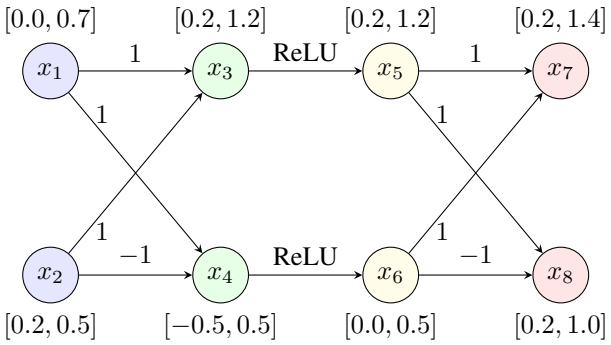


图 1. 神经网络示例

在仅靠传播无法确定某个属性不必要时发挥作用。一旦确定某个属性不必要，我们就会恢复原始界，然后继续处理下一个属性。

IV. 我们的方法

本节介绍我们结合使用的两种方法，以提高神经网络基于逻辑的解释的计算效率。首先，我们介绍用于估计神经元取值范围的 Box 方法，并着重说明它在识别可从解释中排除的属性方面所发挥的作用，而这正是我们方法的核心。接下来，我们讨论 Box 方法如何细化神经元界限并帮助简化混合整数线性规划约束，从而进一步提高解释过程的效率。

A. 基于逻辑的可解释人工智能的 *Box* 界传播

Box 方法是一种用于函数的凸松弛，通过区间算术计算下界与上界。随后，可以用此方法计算神经元的界 [15]。为说明该方法，我们考虑图 1 中的简单神经网络。该神经网络由一个输入层、一个隐藏层和一个输出层组成。每层包含两个神经元。该神经网络有两个输入： x_1 ，其范围为 $[0.0, 0.7]$；以及 x_2，其范围为 $[0.2, 0.5]$。为简化示例，我们将隐藏层中的每个神经元拆分为两部分：一部分表示线性变换的输出，另一部分表示 ReLU 激活的输出。线性变换的权重由边上的权重表示。每个神经元的偏置均为 0，为简化示例已省略。在每个神经元的上方或下方，都有一个区间表示该神经元可能取值的紧界。这些初始区间可通过优化获得。请注意，如果 x_5 达到上界 1。2，那么 $x_1 = 0.7$ 和 $x_2 = 0.5$。然而，在此输入下，神经元 x_6 的输出变为 0。2。因此，输入 $x_1 = 0.7$ 和 $x_2 = 0.5$ 会使神经元 x_7达到上界。如果网络包含更多层， x_7 和 x_8 的值将经过 ReLU 激活函数，其结果将传播到后续层。

Box 方法可用于估算神经网络中每个神经元可能取值的范围。然而，由于这种

方法依赖于过近似，因此得到的区间可能比实际取值范围更宽。为了说明 Box 方法的工作原理，并展示它可能产生较宽松的区间，我们从这些输入范围$x_1 = [0.0, 0.7]$ 和 $x_2 = [0.2, 0.5]$开始应用该方法。首先，如公式 (10) 和 (11)-(13) 所示，分别对隐藏层神经元 x_3 和 x_4 进行计算：

$$x_3 = [0.0, 0.7] + [0.2, 0.5] = [0.2, 1.2] \tag{10}$$
$$x_4 = [0.0, 0.7] + -1 \cdot [0.2, 0.5] \tag{11}$$
$$= [0.0, 0.7] + [-0.5, -0.2] \tag{12}$$
$$= [-0.5, 0.5] \tag{13}$$

区间算术用于计算 x_3 和 x_4的取值范围。对于 x_3，分别将区间的下界与上界相加即可。对于 x_4，首先将区间 $[0.2, 0.5]$ 乘以 -1。乘以负数会反转区间，交换下界与上界，并改变各界的符号。接下来，将区间 $[0.0, 0.7]$ 与区间 $[-0.5, -0.2]$相加，方法与处理 x_3时相同：将下界相加，并将上界相加。然后，我们分别对 x_3 和 x_4应用 ReLU 激活函数，它们对应于神经元 x_5和 x_6：

$$x_5 = \text{ReLU}(x_3) = [\text{ReLU}(0.2), \text{ReLU}(1.2)] \tag{14}$$
$$= [0.2, 1.2] \tag{15}$$
$$x_6 = \text{ReLU}(x_4) = [\text{ReLU}(-0.5), \text{ReLU}(0.5)] \tag{16}$$
$$= [0.0, 0.5] \tag{17}$$

对于 $x_3 = [0.2, 1.2]$，两个界均为正，因此 $x_5 = [0.2, 1.2]$。对于 $x_4 = [-0.5, 0.5]$，负的下界会被截断为 0。0，得到 $x_6 = [0.0, 0.5]$。ReLU 函数会将负值设为 0。0，同时保持正值不变。接下来，我们可以计算输出层，它对应于神经元 x_7 和 x_8。确定 x_7 和 x_8 的界时，采用与处理 x_3 和 x_4相同的方法。然后，$x_7 = [0.2, 1.7]$ 和$x_8 = [-0.3, 1.2]$。通过 Box 方法得到的区间比神经元实际可能的取值范围更宽，如图 1 所示。例如，对于神经元x_7，精确区间为 $[0.2, 1.4]$，而 Box 方法得到的区间更宽，为 $[0.2, 1.7]$。在 Box 方法中，采用以下运算进行区间算术，其中 lb_1，ub_1，lb_2，ub_2，$p \in \mathbb{R}$ 和 $p \geq 0$：

$$[lb_1, ub_1] + [lb_2, ub_2] = [lb_1 + lb_2, ub_1 + ub_2] \tag{18}$$
$$-1 \cdot [lb_1, ub_1] = [-ub_1, -lb_1] \tag{19}$$
$$\text{ReLU}([lb_1, ub_1]) = [\text{ReLU}(lb_1), \text{ReLU}(ub_1)] \tag{20}$$
$$p \cdot [lb_1, ub_1] = [p \cdot lb_1, p \cdot ub_1] \tag{21}$$

在我们的工作中，Box 方法的目标是在解释过程中估计区间。该过程与算法 1 类似，即从 $\mathcal{I}$ 中逐个移除属性。对于每个被移除的属性，我们计算使用 Box 方法计算网络中每个神经元的区间。其目的是确定所考虑的属性对于解释是否必要。当对应类别 $\mathcal{M}(\mathcal{I})$ 的神经元下界大于输出层中其他神经元的上界时，Box 方法可以确保某个属性并非必要。在这种情况下，无需使用 MILP 求解器进行检查 $(\mathcal{X} \setminus \{x_i = v_i\}) \cup \mathcal{F} \models E$ 算法 1。因此，Box 方法的目标是在不借助 MILP 求解器的情况下，判定某个属性并非必要。

为此，我们调整算法 1，使其在每次迭代中先使用 Box 方法，再检查是否满足 $(\mathcal{X} \setminus \{x_i = v_i\}) \cup \mathcal{F} \models E$。此外，我们根据 $\mathcal{X} \setminus \{x_i = v_i\}$ 中的值和域约束 $\mathcal{D}$，通过 Box 方法进行传播。对于 $\mathcal{X} \setminus \{x_i = v_i\}$ 中的属性，我们使用实例 $\mathcal{I}$ 的具体值。对于其他属性，我们使用依据 D 得到的区间。为说明这一思路，请看以下两个示例。

示例 *1*：让我们重新考察图 1 中的神经网络。考虑输入值 $x_1 = 0.7$ 和 $x_2 = 0.2$。此时，$x_7 = 1.4$ 和 $x_8 = 0.4$。因此，该神经网络预测的类别对应于神经元 x_7。根据我们对算法 1 的调整，我们首先检查 Box 方法能否确定属性 x_2 对解释而言并非必要。为此，我们从初始区间 $x_1 = [0.7, 0.7]$ 和 $x_2 = [0.2, 0.5]$ 开始。使用 Box 方法，我们得到 $x_7 = [1.1, 1.7]$ 和 $x_8 = [0.4, 1.0]$。由于 x_7 的下界大于 x_8 的上界，Box 方法可以保证属性 x_2 可从解释中移除。因此，无需使用求解器来验证是否满足 $(\mathcal{X} \setminus x_i = v_i) \cup \mathcal{F} \models E$。仅凭 Box 方法就足以确定属性 x_2 对解释而言并非必要，从而避免了调用求解器所带来的计算开销。

例 *2*：现在，考虑算法 1 首先评估属性 x_1。换句话说，我们将先检查 Box 方法能否确定属性 x_1 对于解释而言是否非必要。为此，我们从初始区间 $x_1 = [0.0, 0.7]$ 和 $x_2 = [0.2, 0.2]$开始。使用 Box 方法，我们得到 $x_3 = [0.2, 0.9]$ 和 $x_4 = [-0.2, 0.5]$。接着，$x_5 = [0.2, 0.9]$ 和 $x_6 = [0.0, 0.5]$。最后，Box 方法给出 $x_7 = [0.2, 1.0]$ 和 $x_8 = [-0.3, 0.5]$。在这种情况下，x_7 的下界小于 x_8的上界，这意味着 Box 方法无法保证可以从解释中移除属性 x_1。因此，有必要使用求解器来确定能否从解释中移除属性 x_1。之所以需要这样做，是因为 Box 方法计算出的区间比实际可能的取值范围更宽，仅凭 Box 区间无法确定这一决定。

B. 界传播中的约束简化

我们方法的第二部分着重利用 Box 方法得到的界，帮助求解器更高效地执行可满足性检查。这

适用于 Box 方法无法确定某个属性是否为不必要、因而需要使用 MILP 求解器的情况。在 Box 计算阶段，我们记录该方法计算出的所有神经元的下界与上界。在构建混合整数线性规划模型时（可在神经网络训练完成后立即进行），我们确定其神经元的紧下界与上界。此步骤在计算解释之前执行。模型构建期间计算出的这些紧界考虑了域约束 $\mathcal{D}$。计算解释时，Box 方法会将属性在 ($\mathcal{X} \setminus \{x_i = v_i\}$) 中的特定输入值纳入计算。因此，由于 Box 方法使用输入属性的特定输入值，而非区间，可能会得到比先前计算出的界更紧的界。在大多数情况下，这使 Box 方法能够计算出更窄的界，从而进一步收紧界。

举例来说，再次考虑例 2。Box 方法无法保证属性 x_1 不能从解释中移除。因此，我们必须使用 MILP 求解器进行检查。不过，我们可以利用 Box 方法找到的信息。考虑图 1 中的范围。在该例中，应用 Box 方法之前，各神经元的范围为 $x_3 = [0.2, 1.2]$, $x_4 = [-0.5, 0.5]$, $x_5 = [0.2, 1.2]$,$x_6 = [0.0, 0.5]$, $x_7 = [0.2, 1.4]$, $x_8 = [0.2, 1.0]$。应用 Box 方法后，界变得更窄：$x_3 = [0.2, 0.9]$,$x_4 = [-0.2, 0.5]$, $x_5 = [0.2, 0.9]$, $x_6 = [0.0, 0.5]$, $x_7 =[0.2, 1.0]$, $x_8 = [-0.3, 0.5]$。请注意，对于神经元 x_8，在调用求解器之前，我们可以将其界更新为 $[0.2, 0.5]$ 。这是因为，仅依据域约束 $\mathcal{D}$计算出的紧界，已确保 x_8 的下界为 0.2。其余界则保留 Box 方法确定的值。此外，为了说明 MILP 约束如何根据识别出的新界发生变化，请考虑公式 6 中与神经元 x_5相关的约束。该约束最初为 $x_5 \leq 1.2z_5$。然而，利用从 Box 方法获得的更新后上界，现在可以将其细化为 $x_5 \leq 0.9z_5$。同样，当识别出更紧的下界时，也可以细化公式 4 中的约束。神经元范围的这些缩减，或者说 MILP 模型中变量范围的缩减，有助于求解器加快解释的计算。

因此，利用 Box 方法得到的界，简化过程会将 Box 计算出的下界和上界与初始确定的界进行比较。根据比较结果，我们保留最大的下界和最小的上界，从而有效缩小并细化范围，得到更紧的界。此外，如果 Box 方法为神经元 x_j^l 找到的下界大于零，我们可以将公式 (4)-(8) 中的约束简化为 $x_j^l = \sum_{i=1}^{n_{l-1}} w_{j,\ i}^l x_i^{l-1} + b_j^l$。另一方面，如果上界小于或等于零，我们可以将这些约束简化为 $x_j^l = 0$。这两种情况下，我们都会丢弃一个二元变量 z_j^l。由于 MILP 求解器通常在二元变量较少时效率更高，这种精简有望提升计算效率。完成这些简化后，我们求解一个混合整数线性规划问题，以确定能否将该属性排除在解释之外。使用求解器确定该属性是否应保留在解释中后，我们会丢弃通过 Box 方法得到的界。随后，在处理下一个属性之前，我们恢复原始界。Box 方法和简化步骤会针对每个属性独立执行。这确保计算专门针对当前正在评估的 $\mathcal{I}$ 子集相关的约束和界进行。

V. 实验结果

为评估我们方法的性能，我们在 UCI 机器学习资料库[1]中的五个数据集上开展了实验：鸢尾花、葡萄酒、声呐、数字和 MNIST。这些数据集涵盖了不同的复杂程度，涉及属性数量和类别数量。鸢尾花是最简单的数据集，包含 4 个属性和 3 个类别；MNIST 则最为复杂，包含 784 个属性和 10 个类别。

针对每个数据集，我们训练了多个深度和宽度各异的神经网络架构。测试配置包括 3 至 8 层的网络，每层包含 3 至 16 个神经元，具体取决于数据集。我们的目标是评估解释性能如何随模型复杂度的提高而变化。由于使用 MILP 求解器会带来较高的计算复杂度，我们重点研究小型至中型神经网络。这些神经网络使用 TensorFlow 库实现并训练。训练过程采用标准的 90%-10% 数据划分，其中 90% 的数据用于训练，10% 用于测试。我们使用 批量大小 8，并对每个神经网络训练 100 训练轮次。

我们比较了两种解释方法：INMS（由 Ignatiev、Narodytska 和 Marques-Silva [6] 提出）以及我们提出的方法，该方法结合了界传播和约束简化。两种方法得出的解释相同，从而确保了正确性和最小性。评估采用了多项性能指标，包括在多个实例上计算解释所需的总时间、混合整数线性规划求解器生成解释所花费的总时间、神经元界限得到收紧的百分比，以及被丢弃的二元变量百分比。神经元界限得到收紧的百分比按如下方式计算：对于每个实例和每个属性，我们首先应用 Box 方法，按照算法 1 所述，判断是否可以安全地将该属性从解释中移除。如果 Box 方法未能奏效，我们就统计神经网络中的神经元总数，以及界限得到收紧的神经元数量。由于不同实例可能需要调用混合整数线性规划求解器不同次数，因此神经元会被多次计数，并且每次都会检查其界限是否得到收紧。最终百分比通过汇总所有实例的计数得出。被丢弃的二元变量百分比也采用类似的计算方法。

[1]https://档案.ics.uci.edu/ml/

表 I INMS 与我们的方法的比较，重点展示总解释时间、求解器执行时间，以及界传播和约束简化对神经元界限和二元变量缩减的影响。

数据集	NN	解释时间（秒） INMS	解释时间（秒） Ours	求解器（秒） INMS	求解器（秒） Ours	% 界收紧后	% 二元变量移除（之前）	% 二元变量移除（我们的方法）
Iris (4)	3L-3N	**0.92**	1.15	0.24	0.05	92.83	0.00	31.61
	6L-6N	**1.28**	1.79	0.56	0.05	96.04	21.21	56.43
Wine (13)	3L-4N	**1.71**	2.72	0.86	0.06	100.00	0.00	46.90
	6L-8N	**4.77**	5.72	3.84	0.06	97.33	23.25	67.08
声呐 (60)	3L-4N	**6.09**	13.20	4.88	0.06	100.00	10.00	65.13
	6L-8N	**16.53**	31.75	15.11	0.07	99.71	19.03	51.57
	8L-12N	160.78	**58.52**	158.94	0.07	95.61	8.13	48.19
数字 (64)	3L-8N	**186.55**	189.13	174.20	0.58	100.00	0.00	52.13
	4L-12N	623.03	**287.15**	609.17	0.56	100.00	0.00	64.67
	4L-16N	1510.58	**350.66**	1494.95	0.56	100.00	0.00	68.44
	6L-12N	3505.43	**375.80**	3489.62	0.58	99.61	5.71	66.65
MNIST (784)	4L-8N	**1455.59**	2443.89	1440.52	0.13	100.00	0.00	61.41
	4L-10N	5267.60	**2899.72**	5249.62	0.13	100.00	0.00	62.59
	6L-8N	9170.63	**2513.24**	9152.08	0.13	100.00	0.00	58.76

如果 Box 方法确定下界大于零或上界小于或等于零，则可以简化公式 (4)–(8) 中的约束，在这两种情况下都能有效消除二元变量 z_j^l。这一指标的动机在于，混合整数线性规划求解器通常在二元变量较少时表现更好，因此这种精简可以提高计算效率。

我们从测试数据中随机选取实例生成解释，以确保两种方法之间的一致性。各数据集分析的实例数量如下：鸢尾花（15）、葡萄酒（18）、声呐（21）、数字（180）和 MNIST（20）。此外，在算法 1 的蕴涵检查中，我们通过 DOcplex API 使用 IBM-CPLEX[2]，对混合整数线性规划约束进行建模和处理。所有实验细节均可在我们的代码仓库中查看[3]。所有实验均在一台 MacBook Pro 上运行，该设备配备 2.60 GHz Intel Core i7 处理器和 16 GB 内存，运行 macOS 13.3。

A. 结果

表 I 展示了针对每个数据集收集的数据。如**解释时间（秒） INMS** 和**解释时间（秒） 我们的方法**两列所示，有少数情况下我们的方法耗时比 INMS 更长。这可能是因为 Box 方法引入了额外开销，例如检查界和细化约束；对于较简单的模型，这些开销可能会抵消其潜在收益。然而，我们观察到，随着神经网络复杂度增加（例如增加层数或神经元数量），与原始方法相比，我们的方法显著缩短了解释总时间。改进的原因之一是 Box 方法效率较高，在某些情况下能够识别出冗余属性，从而无需调用 MILP 求解器并降低计算成本。另一个原因是简化过程会细化神经元的上下界，通常使界限范围变窄。这使求解器能够更高效地运行。

在 Digits 数据集中，最简单的神经网络在 INMS（186.55s）与我们的方法（189.13s）之间的耗时差异微乎其微。然而，对于最复杂的神经网络（6 层、12 个神经元），我们的方法将解释时间从 3505.43s 缩短至 375.80s，性能提升超过 89.26%。在 MNIST 数据集中，对于最复杂的神经网络（6 层、8 个神经元），我们的方法将总解释时间从 9170.63s（INMS）缩短至 2513.24s。该案例涉及此数据集中最复杂的神经网络，其属性数量也是我们实验中最多的，凸显了我们方法的有效性。

现在，我们来探讨有助于更好地理解我们的方法为何以及如何取得优异表现的结果。首先，我们关注 MILP 求解器生成解释所花费的总时间，如 **求解器（秒）INMS** 和 **求解器（秒）我们的方法**列所示。值得注意的是，我们观察到我们的方法带来了显著改进。例如，对于 MNIST 上每层有 8 个神经元、共 6 层的神经网络，原方法的求解器计算时间总计为 9152.08 秒，而我们的方法将其缩短至仅 0.13 秒。虽然我们的方法还涉及一些额外步骤，例如计算界和细化约束，但与求解器计算时间的大幅缩短相比，这些任务带来的额外开销极小。这凸显了我们的方法的明显优势，也证明了整体性能提升的合理性。关于被收紧的神经元界限所占比例，我们的方法始终收紧了 90% 以上，在某些情况下甚至达到 100%，如**界收紧比例（%）**列所示。大量神经元界限得到收紧，有助于提升求解器可扩展性，这一点体现在解释总时间的大幅缩短上。

最后，我们讨论解释过程中被移除的二元变量百分比。为了进行更全面的比较，我们还纳入了原始

[2]https://www.ibm.com/br-pt/analytics/cplex-optimizer [3] https://github.com/ronaldogomes96/explication-ann/tree/dev/ronaldo

INMS 方法，该方法可以在生成解释之前，根据输入界简化二元变量。通过这种方式，我们可以评估我们的方法是否能够在整个解释过程中舍弃更多二元变量。如**移除的二元变量百分比（之前）**和**移除的二元变量百分比（我们的方法）**两列所示，我们的方法在所有数据集和神经网络配置中，始终移除了超过 50% 的二元变量。此外，在每种情况下，与解释过程开始前的精简结果相比，移除的变量比例都有所提高。这些结果凸显了我们的方法在进一步简化混合整数线性规划公式方面的有效性，从而提升了求解器效率和可扩展性。

VI. 结论

在本研究中，我们提出了一种改进的神经网络解释方法，该方法集成了界传播和约束简化，以提高计算效率。我们将该方法与 Ignatiev 等人提出的原始方法进行了比较。该原始方法引入了神经网络的基于逻辑的解释，是文献中较为成熟的方法。

实验结果表明，随着神经网络复杂度的增加，我们方法的优势变得更加显著。例如，在 Digits 数据集上，最复杂神经网络的解释时间从 3505.43 秒降至仅 375.80 秒；在 MNIST 数据集上，即使对于最复杂的神经网络，求解器时间也从超过 9150 秒降至 0.13 秒。通过持续收紧超过 90% 的神经元界限并移除超过 50% 的二元变量，我们的方法能够在大规模场景下简化解释任务。这些结果凸显了两点重要认识。首先，我们的方法能够高效识别并消除无关的输入属性，从而减少混合整数线性规划查询的数量。其次，神经元界限的细化简化了混合整数线性规划问题，从而显著提升了求解器速度。

这些发现为未来研究指明了若干有前景的方向。其中之一是探索其他传播界的方法，例如 Zonotope 和 DeepPoly [16, 17]，以处理规模更大的神经网络。与 Box 方法相比，这些方法生成的界更精确，但代价是计算复杂度更高。此外，鉴于我们的方法与混合整数线性规划求解器紧密集成，未来还可以通过定制求解器，使其能够利用本方法生成的精化后的约束结构，从而进一步改进。

参考文献

[1] I. Goodfellow、Y. Bengio 和 A. Courville，《深度学习》。MIT 出版社，2016 年。[2] P. W. Koh 和 P. Liang，“通过影响函数理解黑箱预测”，载于第 *34* 届 *ICML*，2017 年。[3] M. T. Ribeiro、S. Singh 和 C. Guestrin，“我为什么应该信任你？：解释任何分类器的预测”，载于第 *22* 届 *KDD*，2016 年。[4] ——，“Anchors：高精度、与模型无关的解释”，载于第 *32* 届 *AAAI*，2018 年。

[5] A. Shih、A. Choi 和 A. Darwiche，“解释贝叶斯网络分类器的符号方法”，载于第 *27* 届 *IJCAI*，2018 年。[6]A. Ignatiev、N. Narodytska 和 J. Marques-Silva，“机器学习模型的基于溯因的解释”，载于第 *33* 届 *AAAI*，2019 年。[7] Y. Izza 和 J. M. Silva，“使用可满足性问题解释随机森林”，载于第 *30* 届 *IJCAI*，2021 年。[8] N. Gorji 和 S. Rubin，“存在域约束时分类器决策的充分理由”，载于第 *36* 届 *AAAI*，2022 年。[9] E. Wang、P. Khosravi 和 G. Van den Broeck，“概率充分解释”，载于第 *30* 届 *IJCAI*，2021 年。[10] S. Bassan 和 G. Katz，“迈向形式化可解释人工智能：神经网络的形式化近似最小解释”，载于第 *29* 届 *TACAS*，2023 年。[11]G. Audemard、J.-M. Lagniez、P. Marquis 和 N. Szczepanski，“提升树的溯因解释计算”，载于第 *26* 届 *AISTATS*，2023 年。[12] F. M. Rocha Filho、T. A. Rocha、R. P. F. Ribeiro 和 A. R. Rocha Neto，“带拒绝选项的线性支持向量分类器的基于逻辑的解释”，载于第 *12* 届 *BRACIS*，2023 年。[13] H. Lakkaraju、N. Arsov 和 O. Bastani，“稳健且稳定的黑箱解释”，载于第 *37* 届 *ICML*，2020 年。[14]C. Lin 、 I. Covert 和 S.-I. Lee，“论基于移除的特征归因的鲁棒性”，载于第 *37* 届 *NeurIPS*，2023 年。[15]S. Wang、K. Pei、J. Whitehouse、J. Yang 和 S. S. Jana，“使用符号区间对神经网络进行形式化安全分析”，载于第 *27* 届 *USENIX*，2018 年。[16] T. Gehr、M. Mirman、D. Drachsler-Cohen、P. Tsankov、S. Chaudhuri 和 M. T. Vechev，“AI2：使用抽象解释进行神经网络安全性与鲁棒性认证”，载于第 *39* 届 *IEEE SP*，2018 年。[17] G. Singh、T. Gehr、M. P¨uschel 和 M. T. Vechev，“用于神经网络认证的抽象域”，载于第 *46* 届 *POPL*，2019 年。[18] S. Wang、H. Zhang、K. Xu、X. Lin、S. S. Jana、C.-J. Hsieh 和 Z. Kolter，“Beta-CROWN：通过逐神经元分割约束进行高效界传播，以实现神经网络鲁棒性验证”，载于第 *35* 届 *NeurIPS*，2021 年。[19] H. Zhang、S. Wang、K. Xu、L. Li、B. Li、S. S. Jana、C.-J. Hsieh 和 J. Z. Kolter，“基于界传播的神经网络验证的通用割平面”，载于第 *36* 届 *NeurIPS*，2022 年。[20] D. Kroening 和 O. Strichman，《判定过程》。施普林格，2016 年。[21] M. B´enichou、J.-M. Gauthier、P. Girodet、G. Hentges、G. Ribi`ere 和 O. Vincent，“混合整数线性规划实验”，《数学规划》，第 1 卷，1971 年。[22]V. Tjeng、K. Y. Xiao 和 R. Tedrake，“使用混合整数规划评估神经网络的鲁棒性”，载于第 *7* 届 *ICLR*，2019 年。