分类号 TP311.5 学校代码 10590
UDC 004 密 级 公开

深圳大学硕士学位论文

多路径方法在神经网络验证中的研究与应用

郑烨

学位类别	电子信息硕士学位
专业名称	软件工程
学院（系、所）	计算机与软件学院
指 导 教 师	刘嘉祥

深圳大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：所呈交的学位论文多路径方法在神经网络验证中的研究与应用，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名：　　日期：　　年　　月　　日

学位论文使用授权说明

（必须装订在印刷本首页）

本学位论文作者完全了解深圳大学关于收集、保存、使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属深圳大学。学校有权保留学位论文并向国家主管部门或其他机构送交论文的电子版和纸质版，允许论文被查阅和借阅。本人授权深圳大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（涉密学位论文在解密后适用本授权书）

论文作者签名：　　导师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

摘　要

神经网络已广泛应用于各种现实场景，但由于自身不可避免的过拟合以及攻击技术的发展，使得无论来自于自然或人为影响，神经网络都是极其不稳定的。因此，在安全攸关的应用场景下，神经网络的鲁棒性需要得到严格验证。基于线性近似的界限传播方法在神经网络验证中具有重要地位，此类方法能够在合理时间内得到可接受的验证精度，因此被广泛应用。如何在不增加时间代价的情况下，提高此类方法的验证精度，是一个备受关注的问题。关于此问题，本文做出以下贡献：

（一）针对各种界限传播方法，本文提出界限传播路径的概念。在此概念下，现有界限传播方法可看作仅使用单条界限传播路径，是这一概念的特例。使用多条界限传播路径可以有效改善这类方法的精度。本文形式化地说明了使用多条界限传播路径的界限传播方法，并从理论上证明了其可靠性和验证精度效果。

（二）为了提高实用性和验证速度，本文将多路径界限传播方法实现到机器学习框架 PyTorch 上，并开源为 MpBP 工具。MpBP 能够在 GPU 上并行化多条界限传播路径，因此在与传统界限传播方法相当的时间内显著提高了验证精度。MpBP 能够处理 Tiny ImageNet 规模的数据集网络，同时它支持卷积神经网络结构和类 PyTorch 脚本用法，具有较高的易用性，完备了“训练—验证”流程。

综上，本文提出界限传播路径的概念，将各种界限传播方法扩展到其对应的多路径界限传播方法；并将多路径界限传播在 PyTorch 框架上并行化，开发了高效而易用的鲁棒性验证工具。

关键词：形式化验证；神经网络验证；神经网络鲁棒性；界限传播

Abstract

Neural networks have been widely used in various realistic scenarios. However, due to the inevitable overfitting and the development of attack methods, they are extremely unstable, no matter from natural or human influence. Therefore, the robustness of neural networks needs to be formally verified in many safety-critical application scenarios. Bound propagation methods have an important role in neural network verification. This kind of methods can obtain acceptable verification accuracy at a reasonable time cost, making them widely used. How to improve the verification accuracy of this kind of methods without increasing the time cost is a problem that received much attention. Regarding this research topic, this thesis makes the following contributions:

(1) This thesis proposes the notion of bound propagation path for bound propagation methods. Under this notion, existing bound propagation methods can be regarded as only using a single bound propagation path, which is a special case of the proposed multi-path bound propagation. The accuracy of bound propagation methods can be improved effectively by employing multiple bound propagation paths. In this thesis, a general bound propagation method using multiple propagation paths is formalized, and its soundness and verification accuracy advantage are proved formally.

(2) In order to improve practicability and verification speed, this thesis implements the multi-path bound propagation method on PyTorch, a widely-used machine learning framework, and makes the implementation as an open-source tool: MpBP. MpBP parallelizes multiple bound propagation paths on GPUs, thus significantly improves verification accuracy in comparable time as the traditional bound propagation methods. MpBP can handle large networks in the Tiny ImageNet dataset. Meanwhile, it supports convolutional neural network structure and PyTorch script usage, which enables high usability and a “training-verification” process.

In conclusion, this thesis proposes the notion of bound propagation path; extends bound propagation methods to multi-path bound propagation methods. It also

parallelizes multi-path bound propagation using the PyTorch framework; and develops a efficient and user-friendly engineering implementation.

Keywords: formal verification; neural network verification; robustness of neural networks; bound propagation

目　　录

符号表

符号	含义
$x_{i,j}$	神经网络中第 i 层的第 j 个节点
d_i	神经网络中第 i 层的节点数
$l_{i,j}, u_{i,j}$	节点 $x_{i,j}$ 的数值下界和数值上界
λ	ReLU 函数上近似所使用的下界直线的斜率
m	多路径界限传播中的第 m 条路径
$a_{i,j}^{\geq}, a_{i,j}^{\leq}$	抽象元素，表示节点 $x_{i,j}$ 关于上一层节点的符号下界和符号上界约束
$\gamma^{\geq}, \gamma^{\leq}$	利用 $a_{i,j}^{\geq}$ 和 $a_{i,j}^{\leq}$ 反向替换到输入层的互递归函数，用于表示反向界限传播过程
$\gamma(a_{i,j})$	用于反向界限传播，$\gamma(a_{i,j}) = \left(\gamma^{\geq}(a_{i,j}^{\geq}), \gamma^{\leq}(a_{i,j}^{\leq})\right)$ 为节点 $x_{i,j}$ 关于输入层表示的界限函数
$\gamma^{\star}$	具象函数，用于将界限函数具体化为数值上下界
$low_{i,j}, up_{i,j}$	前向界限传播中节点 $x_{i,j}$ 的界限函数

第 1 章 绪论

1.1 研究背景和意义

神经网络已广泛应用于各种现实场景。过去难以处理的高维复杂问题，如图像分类，神经网络在一定程度上已经可以达到人类的水平。随着硬件性能提升和算法的进步，神经网络模型甚至在一些问题上已经十分成熟，但同时其可解释性[1]却十分有限。另一方面，由于自身不可避免的过拟合以及攻击技术的发展，使得无论来自于自然或人为影响，神经网络都是极其不稳定的。因此，在一些安全攸关的应用场景下，神经网络的部署应用存在较强的局限性。比如对于自动驾驶系统，路标的识别错误可能会导致灾难性的后果。若将神经网络应用于这样的场景，其鲁棒性必须有严格的保证。

具体而言，神经网络的鲁棒性通常指其局部鲁棒性，即针对一个具体输入的鲁棒性。在实际问题中，比如图像分类问题，给定一张具体的输入图片和一个最大扰动范围，若扰动后的图片分类标签不发生改变，称这个分类网络关于输入图片和扰动范围是鲁棒的；反之若存在分类标签的改变，则称这个分类网络不鲁棒。

例如，在自动驾驶系统中，路标识别作为尤其重要的一项任务，识别结果对于光照条件、相机角度、天气变化，以及路标本身的磨损都应具有一定程度的鲁棒性。在图 1-1 中，对于一张“停止（STOP）”路标，不改变其语义的一定大小扰动（如明暗变化）所产生的任意图像都不应被一个自动驾驶神经网络识别为 80 km/h 的限速路标。在另一些场景下，如语音识别，一条刹车语音指令，在不改变其语义的噪声下也不应被识别为直行指令。

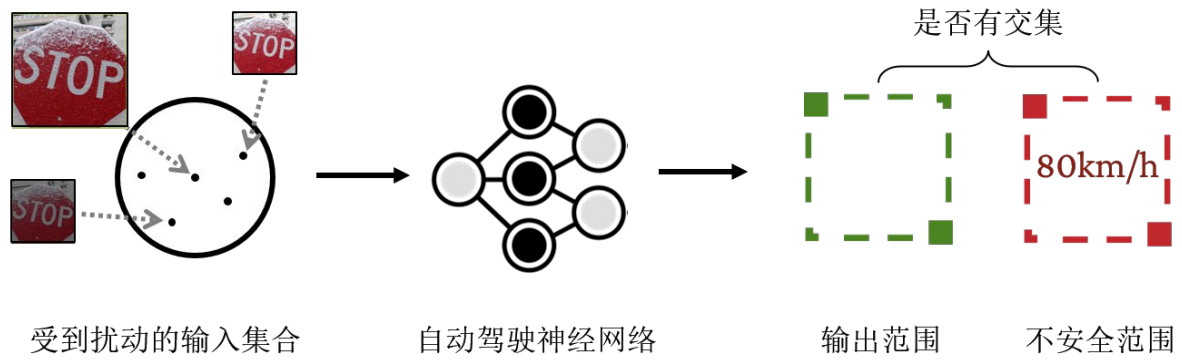


图 1-1 神经网络验证问题

显然，求解上述问题需要得到给定网络的可达集信息，即给定一个神经网络及受到扰动的输入范围，计算输入范围经过这个网络能够到达的输出范围。一种直接的方法是使用足够多的测试用例来进行测试[2]，但有限的测试用例并不能得到准确的可达范围，故不能保证其鲁棒性。面对上述问题，神经网络的形式化验证[3,4]受到了广泛关注，它能够为神经网络的安全性提供严谨的数学保证。

目前神经网络的形式化验证方法可以分为两类。一类方法提供可靠（sound）且完备（complete）的结果，即能够给出准确的“安全”或“不安全”答案。这种方法通常需要刻画给定网络的非线性映射关系，具有较高的计算复杂度，应用规模有限。另一类方法提供可靠却不完备的结果，即可能给出“未知”的答案。这类方法通常对给定网络的非线性结构做上近似处理，具有相对较低的计算复杂度，能够应用于较大规模的网络。

本文关注神经网络验证方法中的界限传播方法（bound propagation）[5]，它是一种可靠但不完备的方法。界限传播能够作为独立的验证方法，也能够结合完备的验证方法，如分支定界（branch and bound，简称 BaB）[6]，作为其定界方法。界限传播方法具有较低的时间复杂度，因此其精度的提高是一个备受关注的目标。本文提出界限传播路径的概念，在此概念下，已有界限传播方法仅使用一条界限传播路径，使用多条界限传播路径能够显著提高现有方法的验证能力。验证能力的提高使得能够更加清晰地把握神经网络的行为，对于神经网络的应用具有一定意义。

从学术的角度，本文着眼于界限传播这种高效的神经网络验证方法。对于界限传播方法，目前的主要关注点在于提高其验证精度。本文在界限传播方法的框架内将其扩展到多路径界限传播，以提高这种方法的验证精度。对于神经网络验证领域有一定的学术价值。

从应用的角度，在一些安全攸关的应用场景下，神经网络的部署应用目前存在较强的局限性。本文提高了现有神经网络验证方法的验证能力，开发了具备较好易用性的验证工具 MpBP，一定程度上能够缓解这一应用局限性，有助于推进神经网络在安全攸关的应用场景下的部署应用。

1.2 国内外研究现状

关于神经网络的安全性的保证，除了能够提供数学保证的验证方法外，测试或基于假设检验的概率方法也能提供一定程度的经验保证，这些方法同样受到广泛关注。

对抗攻击（adversarial attack）方法[2]试图在受到扰动的输入范围内搜索一些反例（counter-example），即神经网络预测结果与输入语义不相符的输入样本。如果找到这样的反例，则可说明给定网络在此受到扰动的输入范围内不足够安全。典型的反例搜索方法包括随机采样和随机梯度下降[7]，也有一些工作使用对抗生成网络（GAN）来生成“较好”的反例[8]。这些方法足够高效，以至于广泛地被用于证明神经网络的不安全性。与对抗攻击相对，对抗防御（adversarial defense）[9,10]方法致力于提升神经网络针对反例的安全性。其中最为典型的方法是再训练（retraining）[11]，这种方法将反例加入训练集用于更新网络。

相较于证明神经网络的不安全性，证明其安全性则困难很多。如果采用测试或基于假设检验的概率方法，则需要穷举整个输入范围的所有样本。然而受到扰动的输入范围理论上包含无穷多个元素，因此是不现实的。

一些工作提供概率安全性保证，这类方法通常使用针对统计模型的假设检验方法，它们可给出概率近似正确（PAC）保证[12,13]，即给定神经网络在一定置信度下，以一定概率满足安全性质。此类方法的不足之处在于，它们是建立在无穷集样本空间上的概率安全性，倾向于给出较为乐观的结果，而不能提供绝对的安全保证。

形式化验证方法关注神经网络在范围输入约束下的数学模型[3]，而非像上述方法一样逐次分析单个输入样本。这使得形式化验证方法能够准确描述神经网络在无穷集输入下的行为，因此能够给出准确的安全性答案。换言之，如果已经通过神经网络验证方法确定一个受到扰动的输入范围是安全的（等价于证明不存在反例），那么上述测试方法或其他攻击方法在任意时间里无法得到一个反例。

目前神经网络验证方法大致可分为两类。一类方法提供可靠且完备的结果，即能够给出且仅会给出安全或不安全答案。显然，这要求计算网络在给定输入范围下的准确可达集。一些文献[14]证明了这类方法具有 NP-hard 时间复杂度。对于最常见的 ReLU 激活函数网络，按照使用的具体技术不同，这类方法可进一步分为三类：（1）基于 SMT 求解[14–16]，在实现上将验证问题编码为可满足性问题来求解，如最常用的 ReLU 函数可用 SMT 语法中的 if-then-else 语句编码；（2）基于混合整数线性规划（mix integer LP），在实现上将验证问题编码为 MILP 问题[17,18]，其中 ReLU 函数的激活与不激活状态分别对应整数决策变量的 1 和 0，从而能够使用 Gurobi[19]等支持 MILP 的求解器求解；（3）基于分支定界法，ReLU 网络的完备验证自然地对应于分支定界法，因为能够对激活状态不确定的 ReLU 节点进行分支。其中第一类基于 SMT 求解的最早代表

方法有 Reluplex[14]以及 Planet[16]，后者在 Reluplex 的基础上对 ReLU 节点做 LP（linear programming）近似以帮助确定更多 ReLU 节点的激活状态。第三类分支定界法在实际应用中，有多种启发式策略确定待分支的激活状态不确定节点，从而加速验证。如 BaBSR[6]利用对偶问题的特征为每个激活状态不确定的节点设计了一种评分规则。另外目前也有工作使用图学习来指导节点的分支[20]。上述三类方法本质上都是对激活状态不确定的节点进行分情况讨论，所以最坏情况下需要描述能够确定该神经网络的 2^N 个线性函数，其中 N 为 ReLU 节点数量。虽然已有很多高效的启发式策略来缓解这一复杂度，但目前使用这类方法验证大规模网络仍然是不现实的。

在输入维度较小时，基于分割输入域[16]的方法同样可以达到完备验证的目的。此类方法通过反复分割输入域构成验证子问题，在足够小的输入域下，神经网络成为或接近线性函数，这使得可以分别快速地验证这些子问题，直到得到准确的验证结果或者超时。

另一类方法提供可靠却不完备的结果，通常它能够给出安全答案，然而当无法给出安全答案时，则会返回未知，这一结果不同于不安全的结论。此类方法通常对 ReLU 神经网络的可达集做上近似处理，即真正的可达集包含于此类方法计算得到的可达集。因此当此类方法得到的可达集与不安全区域有交集时，并不能说明真正的可达集与不安全区域有交集。按照近似方法不同，此类方法又可大致分为两组：（1）基于半正定规划（semi-definite programming，简称 SDP），将 ReLU 激活函数表示为二次约束，再将得到的验证问题松弛为 SDP 问题[21]；（2）基于线性近似，使用线性近似将 ReLU 激活函数近似为包含其凸包的线性约束区域。线性近似方法是不完备验证方法的主流。其中 LP-ALL[22]是对于单个 ReLU 节点线性近似能够达到的最精确方法，它在每个节点构造并求解 LP 问题。界限传播方法是比 LP-ALL 这种约束求解方法更为高效的一类线性近似方法，能够实际应用于验证更大规模的网络，是本文关注的主要方法。

界限传播[5]是基于线性近似的验证方法中较为高效的一类。在范围输入下，每个节点的取值也是范围而非单值，界限传播即传播每个节点取值范围的上下界，它使得验证过程如同神经网络训练的传播（propagation）过程。作为神经网络验证中速度较快的方法，界限传播除了能够作为独立的验证方法，也广泛地被应用于辅助诸如分支定界等完备方法。界限传播方法已有的代表性工具有 DeepPoly[23]、CROWN[24]、Fast-Lin[12]等，三者在计算上等价。其不足之处在于，实际上在验证过程中它们仅使用一条界限传播路径带来的信息，本文将其扩展到多条界限传播路径，有助于得到更为精确的验

证结果。

值得注意的是，除了验证给定神经网络的安全性外，界限传播方法也常被用于训练安全性较高的神经网络。区间传播训练（IBP-training）[9,11,25]相对于传统的使用反例再训练的对抗训练（adversarial training）方法，能够得到安全性更好的神经网络，而不影响准确度。Zhang 等人随后将区间传播训练扩展到精度更高的 IBP-CROWN [25]训练方法。验证技术的发展，也能够帮助训练安全性更高的神经网络。

本文将在第 5 章给出关于界限传播方法更为详尽的相关工作和技术讨论。

1.3 本文的研究内容和贡献

简而言之，本文关注如何提高界限传播验证方法的验证精度。

界限传播方法可进一步分为反向界限传播、前向界限传播、前向+反向界限传播和区间界限传播[5]，它们使用不同的传播策略计算每个节点取值范围的上下界，在精确度和时间消耗之间进行了不同程度的权衡。其中前向界限传播类似神经网络的推导过程，从输入层至输出层进行一趟遍历；反向界限传播类似神经网络的训练过程，在每层反向传播到输入层，这样做可以得到更精确的验证结果。关于界限传播方法的验证精度问题，本文做出以下贡献：

（一）针对各种界限传播方法，本文提出界限传播路径的概念。在此概念下，现有界限传播方法可看作仅使用单条传播路径，是这一概念的特例。使用多条传播路径可以有效改善这类方法的精度。本文形式化地说明了使用多条界限传播路径的界限传播方法，并从理论上证明了其可靠性和验证精度优势。

（二）为了提高实用性和验证速度，本文将多路径界限传播方法实现到机器学习框架 PyTorch[26]上，并开源为 MpBP 工具（https://github.com/formes20/）。MpBP 能够在 GPU 上并行化多条界限传播路径，因此在与传统界限传播方法相当的时间内显著提高了验证精度。MpBP 能够处理 Tiny ImageNet[27]规模的数据集网络，同时它支持卷积神经网络结构和类 PyTorch 脚本用法，具有较高的易用性，完备了“训练—验证”流程。

综上，本文提出界限传播路径的概念，将各种界限传播方法扩展到其对应的多路径界限传播方法；并将多路径界限传播在 PyTorch 框架上并行化，开发了高效而易用的鲁棒性验证工具。

1.4 本文的组织结构

本文包含六个章节。

第 1 章对本文的研究内容做出总体介绍。首先介绍本文的研究背景和意义，指出神经网络验证方法对于保障神经网络部署的重要性；之后介绍国内外的研究现状；最后简要说明本文的研究内容和贡献。

第 2 章介绍本文所需的若干预备知识。包括本文中神经网络的表示和验证问题、神经网络验证问题的线性近似方法、以及界限传播方法。

第 3 章以最广为使用的反向界限传播方法为例，给出反向界限传播路径的概念，并使用多条反向界限传播路径改善此类方法的验证精度。

第 4 章将第 3 章反向界限传播路径的概念扩展到其他典型的界限传播方法，并开发了鲁棒性验证工具 MpBP。工具 MpBP 集成了四种多路径界限传播方法，并使用 GPU 并行化多条路径。

第 5 章给出关于界限传播方法更详尽的相关工作和技术讨论。

第 6 章总结本文工作。

第 2 章 相关知识

本章介绍后文所需的若干预备知识，包括本文中神经网络模型的表示、验证问题的形式、神经网络验证的线性近似方法、和本文关注的界限传播方法。

2.1 神经网络结构

本文关注激活函数为 ReLU 的前馈神经网络的验证问题。前馈神经网络是节点分层表示的有向无环图，第一层称为输入层，最后一层称为输出层，本文用 $x_{i,j}$ 表示第 i 层的第 j 个节点。为避免引入过多符号，在不产生歧义的上下文中也用 $x_{i,j}$ 表示对应节点的值。除了 $i=1$ 即输入层外，每个节点 $x_{i,j}$ 被它的上层节点用单向边连接，这些单向边的权重构成节点 $x_{i,j}$ 的权重向量 $\mathbf{w}_{i,j}$，除权重向量之外每个 $x_{i,j}$ 有一个常量偏移 $b_{i,j}$。$\mathbf{x}_i$ 表示第 i 层节点取值组成的向量。为了描述方便，本文将 ReLU 层作为一种层类型加到每个仿射变换层之后，形成仿射变换层和 ReLU 层交替的神经网络结构，如图 2-1 所示。本文约定除输入层和输出层外，奇数层为 ReLU 层，偶数层为仿射变换层。在这种表示下，一个 ReLU 层节点 $x_{i+1,j}$ 与连接它的仿射变换层节点 $x_{i,j}$ 的关系是 $x_{i+1,j}=\mathrm{ReLU}(x_{i,j})\triangleq\max(0,x_{i,j})$；一个仿射变换层节点 $x_{i,j}$ 与上层节点 $\mathbf{x}_{i-1}$ 的关系是 $x_{i,j}=\mathbf{w}_{i,j}\mathbf{x}_{i-1}+b_{i,j}$。例如在图 2-1 中 $x_{3,1}$ 作为 ReLU 节点有 $x_{3,1}=\mathrm{ReLU}(x_{2,1})$，其中 $x_{2,1}$ 由仿射变换 $x_{2,1}=\mathbf{w}_{2,1}\mathbf{x}_1+b_{2,1}$ 得到。

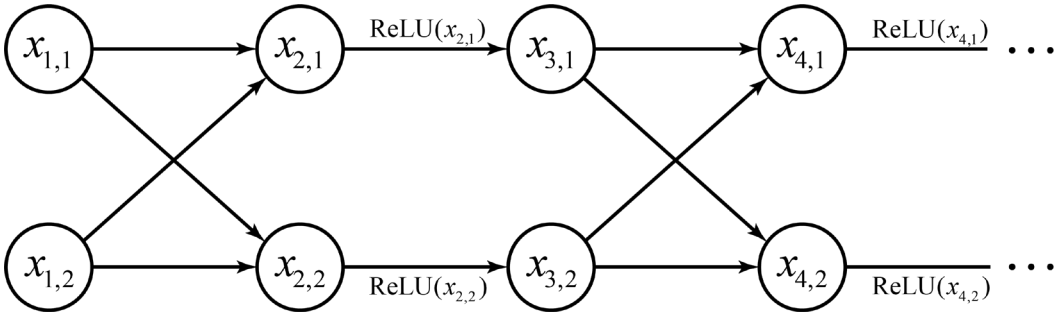


图 2-1 神经网络结构

给定一个有界范围的输入，每个节点 $x_{i,j}$ 的取值不再是一个数值而是一个数值区间 $[l_{i,j},u_{i,j}]$，其中 $l_{i,j}$ 和 $u_{i,j}$ 分别称为节点 $x_{i,j}$ 的数值下界和数值上界。仿射变换层节点 $x_{i,j}$

作为 ReLU 函数的输入，若有 $l_{i,j}<0<u_{i,j}$，则称之后的 ReLU 节点激活状态不确定。

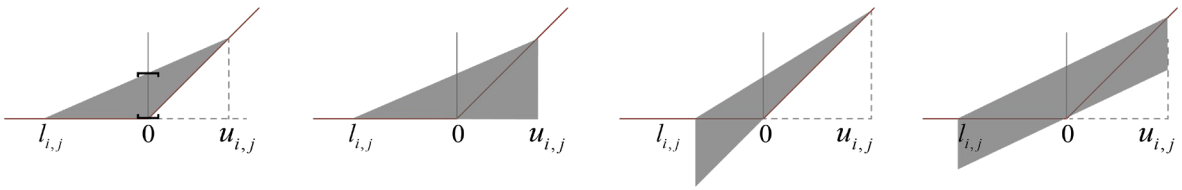
一个神经网络确定了一个从输入到输出的函数 f，从经验上来看 f 通常具有十分复杂的映射关系，因此验证问题最关注的是 f 在输入是一个范围下的可达集性质。

定义 1（验证问题）给定 L 层神经网络 $f:\mathbb{R}^{d_1}\to\mathbb{R}^{d_L}$，其中 d_i 表示第 i 层维数；给定输入范围 $\mathcal{X}_1$，不安全输出范围 $\overline{S}$。验证问题的目标是确定 $f(\mathcal{X}_1)\cap\overline{S}=\varnothing$ 是否成立。

如果可达集 $f(\mathcal{X}_1)$ 与不安全区域 $\overline{S}$ 交集为空，称神经网络 f 关于性质 $(\mathcal{X}_1,\overline{S})$ 是安全的；反之若交集非空，则称关于性质 $(\mathcal{X}_1,\overline{S})$ 是不安全的。例如图像分类问题中，输入域 $\mathcal{X}_1$ 是对一张原始图片经过一定大小扰动的全体图片，不安全区域 $\overline{S}$ 可以是除了正确分类标签之外的其它所有标签，这样上述验证问题即图像分类网络的局部鲁棒性问题[28]。对于分类网络，验证 $f(\mathcal{X}_1)$ 与 $\overline{S}$ 交集为空等价于验证在输入范围 $\mathcal{X}_1$ 下，正确标签对应的节点值恒大于其它标签对应的节点值。因此求解验证问题的关键实际上是求解输出层节点的可达集。对于使用线性近似等可靠但不完备的方法，关键则是如何更准确地近似输出层节点的真正可达集。

2.2 线性近似

神经网络验证中的线性近似方法将非线性激活函数 ReLU 近似为包含其凸包的线性约束区域，这是因为处理线性函数的叠加比处理非线性函数的叠加容易很多。图 2-2 表示常见的四种 ReLU 近似方式。图中的横轴表示一个节点 $x_{i,j}$ 的取值，它的数值上下界 $[l_{i,j},u_{i,j}]$ 跨过原点，因此激活状态不确定；纵轴表示它经过 ReLU 激活函数的取值 $x_{i+1,j}$。蓝色阴影区域表示对 ReLU 函数近似后的输入输出关系。如在图 2-2（a）中 $x_{i,j}=0$ 时，$x_{i+1,j}=\mathrm{ReLU}(x_{i,j})=0$ 被近似为灰色区域与纵轴的交集，即黑色方括号区间。



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

图 2-2 四种 ReLU 近似方式

图 2-2（a） 通常称为 LP 近似[16]，其中 LP 是线性规划（linear programming）的简称。其近似区域重合于 ReLU 激活函数在$[l_{i,j},u_{i,j}]$上的二维凸包，可以表述为以下三个线性约束

$$\begin{aligned} x_{i+1,j} &\le \frac{u_{i,j}}{u_{i,j}-l_{i,j}}(x_{i,j}-l_{i,j}) \\ x_{i+1,j} &\ge x_{i,j} \\ x_{i+1,j} &\ge 0. \end{aligned} \tag{2-1}$$

对单个 ReLU 激活函数的线性近似中，这种近似方法是最精确的。LP 近似的一个重要问题是，每个节点具有两个下界约束，使得确定此节点的数值上下界实际上成为一个最优化问题，这也是其名称 LP 近似的来源。使用 LP 近似计算每个节点的数值上下界能够得到较为精确的结果，但随着网络深度和宽度增加，求解接近输出层的节点数值上下界对应的 LP 问题会变得十分复杂。

图 2-2（b）（c）（d）是更为简单的线性近似，它们分别称为直角三角形近似，钝角三角形近似和平行四边形近似。在这三种近似方法中，每个 ReLU 节点的上界和下界约束分别只有一个，这使得可以利用符号约束逐层地传递计算上下界[29,30]，而不涉及复杂的优化方法。它们是神经网络验证中速度很快的方法。但另一方面，较少的约束意味着较大的精度损失。而且随着网络深度的增加，精度损失的叠加会愈加明显。如何利用这种简单的约束得到尽可能高的验证精度，是目前备受关注的问题。

下面的定理 1 说明这四种近似的可靠性，即$x_{i,j}$经过 ReLU 的值属于经过上述 ReLU 近似的区间。

定理 1 给定$l_{i,j}<0,u_{i,j}>0$，对于任意$\lambda\in[0,1]$及任意$x_{i,j}\in[l_{i,j},u_{i,j}]$，下式成立:

$$\lambda x_{i,j} \le \mathrm{ReLU}(x_{i,j}) \le \frac{u_{i,j}}{u_{i,j}-l_{i,j}}(x_{i,j}-l_{i,j}). \tag{2-2}$$

证明：注意到$u_{i,j}/(u_{i,j}-l_{i,j})\in(0,1)$且$x_{i,j}-l_{i,j}\ge 0$。当$x_{i,j}<0$时，由 ReLU 函数定义，$\mathrm{ReLU}(x_{i,j})=0$，而不等式左边$\lambda x_{i,j}\le 0$恒成立且不等式右边$u_{i,j}(x_{i,j}-l_{i,j})/(u_{i,j}-l_{i,j})\ge 0$，故上式成立；当$x_{i,j}\ge 0$时，$\mathrm{ReLU}(x_{i,j})=x_{i,j}$，而不等式左边$\lambda x_{i,j}\le x_{i,j}$恒成立且$u_{i,j}(x_{i,j}-l_{i,j})/(u_{i,j}-l_{i,j})-\mathrm{ReLU}(x_{i,j})=l_{i,j}(x_{i,j}-u_{i,j})/(u_{i,j}-l_{i,j})\ge 0$，故上式成立。证毕。

取 $\lambda=0$ 及 $\lambda=1$ 时，定理 1 蕴含 LP 近似的可靠性；取 $\lambda=u_{i,j}/(u_{i,j}-l_{i,j})$ 时蕴含平行四边形近似的可靠性。同理可得直角三角形和钝角三角形近似的可靠性。此外定理 1 还表明，任意 $\lambda x_{i,j}$ 都可以作为 $\mathrm{ReLU}(x_{i,j})$ 的下界，从而与上界 $u_{i,j}(x_{i,j}-l_{i,j})/(u_{i,j}-l_{i,j})$ 构成仅有一个线性下界约束和一个线性上界约束的 ReLU 可靠近似。

2.3 符号传播与反向界限传播

Wang 等人[29,30]引入符号传播方法在神经网络验证问题中的应用。目前主流的验证方法使用符号传播方法维护每个节点与输入层节点的约束关系，即用变量表示的约束关系。这可以通过前向传递符号约束得到，也可以反向替换符号约束至输入层得到。

界限传播方法使用激活函数的简单线性近似配合符号传播方法，从而得到给定神经网络中每个节点的界限函数。精确度是衡量界限传播方法验证能力的最重要指标，计算更精确的界限函数意味着能够处理更多的验证问题。根据界限传播策略的不同，界限传播方法可进一步分为反向界限传播（backward bound propagation）、前向界限传播（forward bound propagation）、前向+后向界限传播（forward+backward bound propagation）和区间界限传播（interval bound propagation）[5]，它们在精确度和时间消耗之间进行了不同程度的权衡。

在已有的朴素界限传播方法中，反向界限传播方法具有最高的验证精确度，因此受到广泛关注。对于每个节点 $x_{i,j}$，它维护 $x_{i,j}$ 关于上层节点值 $\mathbf{x}_{i-1}$ 的符号约束，然后通过代入 $\mathbf{x}_{i-1}$ 关于 $\mathbf{x}_{i-2}$ 的符号约束，依次向输入层替换，得到 $x_{i,j}$ 关于输入层 $\mathbf{x}_1$ 表示的上下界符号约束（界限函数）。最后通过代入 $\mathbf{x}_1$ 的数值上下界，得到 $x_{i,j}$ 的数值上下界。替换的过程通常也被称为传播，反向传播到输入层可以得到 $x_{i,j}$ 更精确的上下界。

图 2-3 展示了反向界限传播方法（如 DeepPoly[23]和 CROWN[24]）计算每个节点数值上下界的过程。网络输入范围 $\mathcal{X}_1=\{(x_{1,1},x_{1,2}):-1\le x_{1,1}\le 1,-1\le x_{1,2}\le 1\}$ 且各节点偏移都为 0，节点权重在前向边上标出。注意节点 $x_{4,1}$ 由 $x_{3,1}$ 和 $x_{3,2}$ 经过仿射变换得到，它们之间的关系是 $x_{4,1}\ge x_{3,1}-x_{3,2}$ 及 $x_{4,1}\le x_{3,1}-x_{3,2}$，这是 $x_{4,1}$ 取值的符号约束。为了得到 $x_{4,1}$ 尽可能精确的数值上下界，需要得到它关于输入层的符号表示，因此利用 $x_{3,1}$ 和 $x_{3,2}$ 向输入层反向传播。$x_{3,1}$ 由 $x_{2,1}$ 经过 ReLU 函数得到，假设这里使用直角三角形近似，则有

$x_{3,1} \geq 0$ 以及 $x_{3,1} \leq 0.5x_{2,1} + 1$；同理对于 $x_{3,2}$ 有 $x_{3,2} \geq 0$ 以及 $x_{3,2} \leq 0.5x_{2,2} + 1$。那么 $x_{4,1}$ 的上界是 $x_{4,1} \leq 0.5x_{2,1} + 1 - 0$，下界是 $x_{4,1} \geq 0 - (0.5x_{2,2} + 1)$。按照同样的方法继续向输入层替换，可以得到 $x_{4,1}$ 关于输入层变量 $x_{1,1}$ 和 $x_{1,2}$ 表示的界限函数为 $x_{4,1} \geq -0.5x_{1,1} + 0.5x_{1,2} - 1$ 及 $x_{4,1} \leq -0.5x_{1,1} + 0.5x_{1,2} + 1$，最后可代入 $x_{1,1}$ 和 $x_{1,2}$ 的数值上下界得到 $x_{4,1} \geq -2$ 及 $x_{4,1} \leq 2$。因此 $[-2, 2]$ 是 $x_{4,1}$ 的数值上下界。

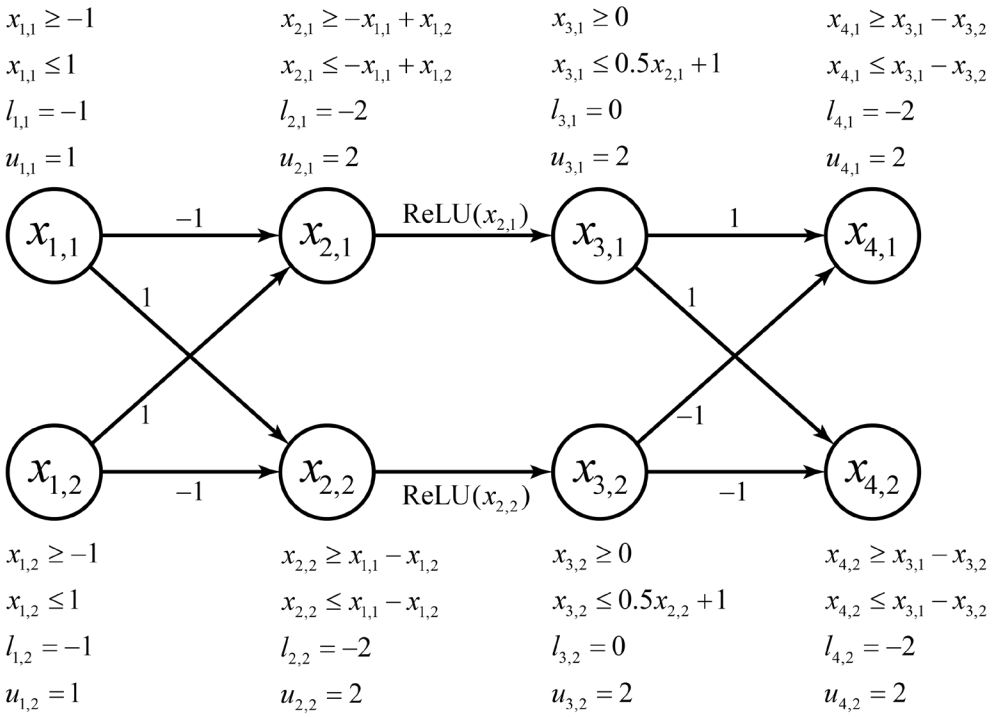


图 2-3 使用反向界限传播计算节点上下界

对于所有非输入层节点，反向界限传播都使用这种替换到输入层的方式计算其数值上下界。除了替换到输入层，实际的工具实现如 DeepPoly 和 CROWN 还使用一个有效的启发式策略来减少对每个 ReLU 激活函数的近似误差：对于激活状态不确定的节点，例如 $x_{2,1}$，如果求得其数值上下界为 $[l_{2,1}, u_{2,1}]$，若 $|l_{2,1}| \geq u_{2,1}$，则选择直角三角形近似，因为此时直角三角形的面积为 $0.5 \times u_{2,1}(u_{2,1} - l_{2,1})$，小于等于钝角三角形的面积 $0.5 \times |l_{2,1}|(u_{2,1} - l_{2,1})$；若 $|l_{2,1}| < u_{2,1}$，则选择钝角三角形近似，因为此时钝角三角形的面积小于直角三角形。更小的面积意味着更小的近似误差。

反向界限传播得到每个节点的数值上下界是可靠的，即对于每个节点，它真正的数值上下界包含于反向界限传播计算得到的上下界。通过这种方法可以快速得到真正可达集的一个上近似，如果一个受到扰动的输入范围被 DeepPoly 或 CROWN 验证是安

全的，那么蕴含它确实是安全的；但反之不一定成立。上近似的精度决定了这种方法的验证能力。

2.4 一维输入下的反向界限传播方法

因反向界限传播方法（以及第 4 章介绍的其他界限传播方法）得到的界限函数是关于输入节点的线性函数，所以在一维输入下，即给定网络仅有一个输入节点时，每个节点的界限函数是一维直线。这意味着界限传播方法在给定输入范围内为每个节点 $x_{i,j}$ 计算一组一维上下界函数，如图 2-4 所示。

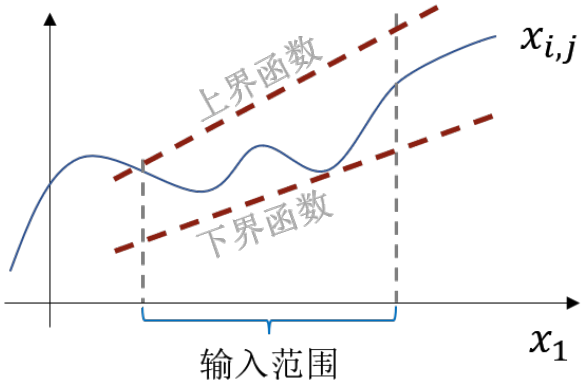


图 2-4 反向界限传播方法在一维输入下的示例

图 2-4 中蓝色曲线表示 $x_{i,j}$ 对应的真实神经网络曲线，它是关于 x_1 的连续非线性函数，准确计算出此函数是 NP-hard 问题。传统界限传播方法在给定输入范围内为 $x_{i,j}$ 找到一组线性上下界函数以确定它上近似的范围，即图中为红色虚线。因线性函数的复合依旧是线性函数，所以每个节点的界限函数可以通过传播（propagation）的方式得到。

下一章将介绍本文的方法，它使用多条反向界限传播路径，能够在给定输入范围内为每个节点 $x_{i,j}$ 计算多组界限函数，从而改善对真正可达集上近似的精度，而仅引入较低的额外时间开销。

第 3 章 多路径反向界限传播

本章以最广为使用的反向界限传播方法为例，给出反向界限传播路径的概念。在此概念下，现有反向界限传播方法可看作仅使用单条传播路径，是这一概念的特例。使用多条传播路径可以有效改善这类方法的精度。本章形式化地说明了使用多条反向界限传播路径的界限传播方法，并从理论和实验上证明了其可靠性和验证精度效果。

本章内容组织如下：第 3.1 节给出反向界限传播路径的形式化描述；第 3.2 节给出一个两条反向界限传播路径的示例；第 3.3 节给出多路径反向界限传播的形式化证明；第 3.4 节在一维输入下给出多路径反向界限传播更为直观的解释；第 3.5 节给出算法描述和复杂度分析；第 3.6 节为实验评估；第 3.7 节为本章小结。

3.1 反向界限传播路径

已有的界限传播方法对每个节点 $x_{i,j}$ 维护一组符号上下界约束。本节将会说明一组符号约束最多构成一条反向界限传播路径，因此它仅能得到 $x_{i,j}$ 的一组界限函数和一组数值上下界。如果对每个节点 $x_{i,j}$ 维护多组符号上下界约束，分别在多组符号约束构成的多条反向界限传播路径上替换到输入层，则可得到 $x_{i,j}$ 的多个数值上下界。不同传播路径上可以使用不同的线性近似产生不同的数值上下界，定理 1 保证这多个数值上下界的交集也是 $x_{i,j}$ 的数值上下界，因此可以得到更精确的上下界，从而进一步构造更精确的上近似。这是本章的主要想法。

给定正整数 n，定义 $[n]=\{1,2,...,n\}$ 为前 n 个正整数构成的集合。$\mathbb{R}_1[\mathbf{x}]$ 表示实系数线性多项式集合。给定 $m\in[M]$，本文用抽象元素 $a_{i,j,m}=(a^{\geq}_{i,j,m},a^{\leq}_{i,j,m})\in\mathcal{A}=\mathbb{R}_1[\mathbf{x}]\times\mathbb{R}_1[\mathbf{x}]$ 表示节点 $x_{i,j}$ 的关于上层节点的符号约束，且限定 $a^{\geq}_{i,j,m}$ 和 $a^{\leq}_{i,j,m}$ 具有以下特定形式

$$\begin{aligned} a^{\geq}_{i+1,j,m},a^{\leq}_{i+1,j,m}&=\sum_{k=1}^{d_i}q_{i+1,k,m}x_{i,k}+r_{i+1,j,m}\\ a^{\geq}_{1,j,m}&=a^{\leq}_{1,j,m}=x_{1,j}. \end{aligned} \tag{3-1}$$

上式中 $q_{i+1,k,m}$ 和 $r_{i+1,j,m}$ 为常数。$a^{\geq}_{i+1,j,m}$ 和 $a^{\leq}_{i+1,j,m}$ 分别是 $x_{i+1,j}$ 关于第 i 层节点表示的线性下界约束和线性上界约束。通过 $x_{i+1,j}$ 依赖的所有节点的符号约束，可以反向传播得到

$x_{i+1,j}$关于输入层表示的界限函数。定义变换函数$\gamma^{\geq},\gamma^{\leq}:\mathbb{R}_1[\mathbf{x}]\to\mathbb{R}_1[\mathbf{x}]$如下

$$\begin{aligned}\gamma^{\geq}(\sum_{k=1}^{d_i}q_{i+1,k,m}x_{i,k}+r_{i+1,j,m})&=\sum_{k=1}^{d_i}\gamma^{\geq}(q_{i+1,k,m}x_{i,k})+r_{i+1,j,m}\\&=\sum_{k=1}^{d_i}\frac{|\delta(q_{i+1,k,m})+1|}{2}q_{i+1,k,m}\gamma^{\geq}(a^{\geq}_{i,k,m})+\sum_{k=1}^{d_i}\frac{|\delta(q_{i+1,k,m})-1|}{2}q_{i+1,k,m}\gamma^{\leq}(a^{\leq}_{i,k,m})+r_{i+1,j,m}\\\gamma^{\leq}(\sum_{k=1}^{d_i}q_{i+1,k,m}x_{i,k}+r_{i+1,j,m})&=\sum_{k=1}^{d_i}\gamma^{\leq}(q_{i+1,k,m}x_{i,k})+r_{i+1,j,m}\\&=\sum_{k=1}^{d_i}\frac{|\delta(q_{i+1,k,m})+1|}{2}q_{i+1,k,m}\gamma^{\leq}(a^{\leq}_{i,k,m})+\sum_{k=1}^{d_i}\frac{|\delta(q_{i+1,k,m})-1|}{2}q_{i+1,k,m}\gamma^{\geq}(a^{\geq}_{i,k,m})+r_{i+1,j,m}\end{aligned}$$

且 $\gamma^{\geq}(x_{1,k})=\gamma^{\leq}(x_{1,k})=x_{1,k}$。上式中$\delta$为符号函数，当$q\geq 0$时，$\delta(q)=+1$；$q<0$时，$\delta(q)=-1$。$\gamma^{\geq}$和$\gamma^{\leq}$为一组互递归函数，它们用于描述$x_{i+1,j}$反向替换传播到输入层的过程，最终得到关于输入层表示的界限函数。在计算$x_{i+1,j}$的下界时，如果$x_{i,k}$对应的系数$q_{i+1,k,m}$为正，需要使用$x_{i,k}$的符号下界；如果$x_{i,k}$对应的系数$q_{i+1,k,m}$为负，则需要使用$x_{i,k}$的符号上界。计算$x_{i+1,j}$的上界时同理。这保证得到的是每个节点的上界和下界。

整个递归过程实际确定了一条反向界限传播路径。具体地，如果定义函数$\gamma(a_{i,j,m})=(\gamma^{\geq}(a^{\geq}_{i,j,m}),\gamma^{\leq}(a^{\leq}_{i,j,m})):\mathcal{A}\to\mathcal{A}$，对任意$m\in[M]$，$\gamma(a_{i,j,m})$确定了使用$x_{i,j}$的第$m$组符号约束计算数值上下界的反向界限传播路径:

$$a_{1,*,m}\leftarrow a_{2,*,m}\leftarrow\ldots\leftarrow a_{i-1,*,m}\leftarrow a_{i,j,m}. \tag{3-2}$$

由M组抽象元素$a_{i,j,m}$可以确定M条反向界限传播路径，通过这M条反向界限传播路径可确定$x_{i,j}$的M个关于输入层表示的界限函数$\gamma(a_{i,j,m})$。定义从$\gamma(a_{i,j,m})$得到数值上下界的具象函数$\gamma^{\star}:\mathcal{A}\to\mathbb{R}\times\mathbb{R}$如下

$$\begin{aligned}\gamma^{\star}\left(\gamma(a_{i,j,m})\right)&=\gamma^{\star}\left(\gamma^{\geq}(a^{\geq}_{i,j,m}),\gamma^{\leq}(a^{\leq}_{i,j,m})\right)\\&=\left(\gamma^{\star}_{\geq}\left(\gamma^{\geq}(a^{\geq}_{i,j,m})\right),\gamma^{\star}_{\leq}\left(\gamma^{\leq}(a^{\leq}_{i,j,m})\right)\right)\end{aligned} \tag{3-3}$$

其中

$$\begin{aligned}\gamma^{\star}_{\geq}\left(\gamma^{\geq}(a^{\geq}_{i,j,m})\right)&=\gamma^{\star}_{\geq}\left(\sum_{k=1}^{d_1}q_{1,k,m}x_{1,k}+r_{1,m}\right)\\&=\sum_{k=1}^{d_1}\frac{|\delta(q_{1,k,m})+1|}{2}q_{1,k,m}l_{1,k}+\sum_{k=1}^{d_1}\frac{|\delta(q_{1,k,m})-1|}{2}q_{1,k,m}u_{1,k}+r_{1,m}.\end{aligned} \tag{3-4}$$

$$\begin{aligned}\gamma_{\le}^{\star}\left(\gamma^{\le}(a_{i,j,m}^{\le})\right)&=\gamma_{\le}^{\star}\left(\sum_{k=1}^{d_1}q_{1,k,m}x_{1,k}+r_{1,m}\right)\\&=\sum_{k=1}^{d_1}\frac{|\delta(q_{1,k,m})+1|}{2}q_{1,k,m}u_{1,k}+\sum_{k=1}^{d_1}\frac{|\delta(q_{1,k,m})-1|}{2}q_{1,k,m}l_{1,k}+r_{1,m}.\end{aligned}\tag{3-5}$$

上式的意义为：对于 $x_{i,j}$ 的数值下界（3-4）式，若 $q_{1,k,m}>0$，使用 $x_{1,k}$ 的下界 $l_{1,k}$；若 $q_{1,k,m}<0$，则使用 $x_{1,k}$ 的上界 $u_{1,k}$。$x_{i,j}$ 数值上界（3-5）式的计算类似。为了简便，记 $\gamma^{\star}(\gamma(a_{i,j,m}))$ 为 $\gamma^{\star}(a_{i,j,m})$。则 $x_{i,j}$ 可计算出 M 个数值上下界 $(l_{i,j,m},u_{i,j,m})=\gamma^{\star}(a_{i,j,m})$。它们的交集也是 $x_{i,j}$ 的数值上下界，取

$$[l_{i,j},u_{i,j}]=\bigcap_{m=1}^{M}[l_{i,j,m},u_{i,j,m}]\tag{3-6}$$

作为节点 $x_{i,j}$ 最终的数值上下界。

抽象元素 $a_{i,j,m}$ 的计算取决于神经网络中 $x_{i,j}$ 的节点类型。对于第 $m\in[M]$ 种近似方式，若 $x_{i,j}$ 是激活状态不确定的节点，定义其抽象元素 $a_{i,j,m}$ 的抽象变换为

$$\mathrm{ReLU}_m^{\#}(a_{i,j,m})=\mathrm{ReLU}_m^{\#}\left((a_{i,j,m}^{\ge},a_{i,j,m}^{\le})\right)=(a_{i+1,j,m}^{\ge},a_{i+1,j,m}^{\le})\tag{3-7}$$

其中 $(a_{i+1,j,m}^{\ge},a_{i+1,j,m}^{\le})$ 表示节点 $x_{i+1,j}$ 对应的抽象元素，具体形式为

$$a_{i+1,j,m}^{\ge}=\lambda_m x_{i,j},\quad a_{i+1,j,m}^{\le}=\frac{u_{i,j}}{u_{i,j}-l_{i,j}}(x_{i,j}-l_{i,j}).\tag{3-8}$$

上式中 $\lambda_m\in[0,1]$ 为常数。即在每条传播路径上对激活状态不确定的节点使用相同的 λ_m。对于激活状态确定的 ReLU 以及仿射变换函数，容易定义其准确的抽象变换。对于激活状态确定的 ReLU 节点的抽象变换，其具体形式为：

$$\begin{cases}a_{i+1,j,m}^{\ge}=a_{i+1,j,m}^{\le}=x_{i,j} & l_{i,j}\ge 0\\ a_{i+1,j,m}^{\ge}=a_{i+1,j,m}^{\le}=0 & u_{i,j}\le 0\end{cases}\tag{3-9}$$

仿射变换节点对应的抽象变换为：

$$a_{i+1,j,m}^{\ge}=a_{i+1,j,m}^{\le}=\mathbf{w}_{i+1,j}\mathbf{x}_i+b_{i,j}\tag{3-10}$$

其中 $\mathbf{w}_{i+1,j}$ 为 $x_{i+1,j}$ 的权重矩阵，$\mathbf{x}_i$ 为第 i 层的节点向量。

已有的反向界限传播方法对每个节点 $x_{i,j}$ 仅维护一组抽象元素 $a_{i,j,1}$，是 $M=1$ 的特例。其 λ_1 对应第 2.3 节所述的启发式策略，本文称它确定的反向界限传播路径为 DeepPoly 路径。使用一条反向界限传播路径仅能得到 $x_{i,j}$ 的一个数值上下界

$[l_{i,j,1},u_{i,j,1}]$。通过维护 $x_{i,j}$ 的多组抽象元素，利用这多组抽象元素构成的多条反向界限传播路径，可得到 $x_{i,j}$ 更多的数值上下界信息。此外，多条路径保证至少有任何一条路径的效果，即至少达到 DeepPoly 的验证精度。

3.2 两条反向界限传播路径的示例

图 3-1 中网络是图 2-3 的扩展，它展示了使用两条反向界限路径计算每个节点数值上下界的过程。其中 $m=1$ 对应 DeepPoly 路径，见图示神经网络外侧第 1~4 行；$m=2$ 对应平行四边形抽象对应的传播路径，见图示神经网络外侧第 5~8 行。网络的输入范围为 $\mathcal{X}_1=\{(x_{1,1},x_{1,2}):-1\le x_{1,1}\le 1,-1\le x_{1,2}\le 1\}$ 且每个节点的偏移为 0 。

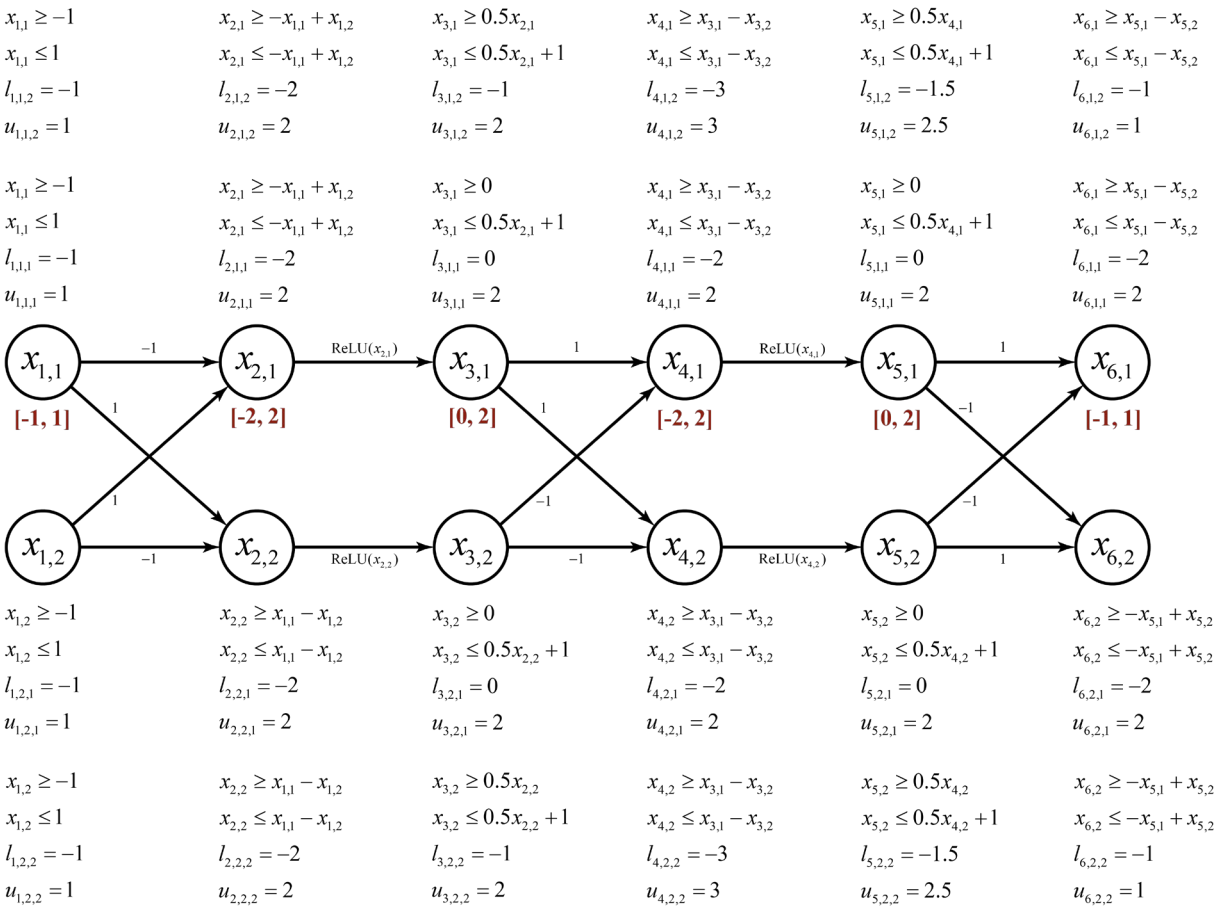


图 3-1 使用两条反向界限传播路径计算节点上下界

对于 ReLU 层节点 $x_{3,1}$，本文维护经过 DeepPoly 近似得到的符号约束 $x_{3,1}\ge 0$ 及 $x_{3,1}\le 0.5x_{2,1}+1$；同时维护经过平行四边形近似得到的符号约束 $x_{3,1}\ge 0.5x_{2,1}$ 及 $x_{3,1}\le 0.5x_{2,1}+1$。这两组符号约束作为 $x_{3,1}$ 的两个抽象元素构成两条不同路径。对于

DeepPoly 路径，其关于输入层的界限函数为

$$\begin{aligned}\gamma(a_{3,1,1}) &= \left(\gamma^{\geq}(a_{3,1,1}^{\geq}), \gamma^{\leq}(a_{3,1,1}^{\leq})\right)\\ &= \left(\gamma^{\geq}(0), \gamma^{\leq}(0.5x_{2,1}+1)\right)\\ &= \left(0, 0.5\gamma^{\leq}(a_{2,1,1}^{\leq})+1\right)\\ &= \left(0, 0.5\gamma^{\leq}(-x_{1,1}+x_{1,2})+1\right)\\ &= \left(0, 0.5\left(-\gamma^{\geq}(a_{1,1,1}^{\geq})+\gamma^{\leq}(a_{1,2,1}^{\leq})\right)+1\right)\\ &= \left(0, 0.5\left(-x_{1,1}+x_{1,2}\right)+1\right).\end{aligned} \tag{3-11}$$

对应的数值下界和上界分别为

$$\begin{aligned}l_{3,1,1} &= \gamma_{\geq}^{\star}\left(0\right) = 0\\ u_{3,1,1} &= \gamma_{\leq}^{\star}\left(0.5\left(-x_{1,1}+x_{1,2}\right)+1\right)\\ &= -0.5l_{1,1}+0.5u_{1,2}+1\\ &= 2.\end{aligned} \tag{3-12}$$

对于平行四边形近似对应的反向界限传播路径，其关于输入层的界限函数为

$$\begin{aligned}\gamma(a_{3,1,2}) &= \left(\gamma^{\geq}(a_{3,1,2}^{\geq}), \gamma^{\leq}(a_{3,1,2}^{\leq})\right)\\ &= \left(\gamma^{\geq}(0.5x_{2,1}), \gamma^{\leq}(0.5x_{2,1}+1)\right)\\ &= \left(0.5\gamma^{\geq}(-x_{1,1}+x_{1,2}), 0.5\gamma^{\leq}\left(-x_{1,1}+x_{1,2}\right)+1\right)\\ &= \left(0.5\left(-\gamma^{\leq}(a_{1,1,2}^{\leq})+\gamma^{\geq}(a_{1,2,2}^{\geq})\right), 0.5\left(-\gamma^{\geq}(x_{1,1})+\gamma^{\leq}(x_{1,2})\right)+1\right)\\ &= \left(0.5\left(-x_{1,1}+x_{1,2}\right), 0.5\left(-x_{1,1}+x_{1,2}\right)+1\right).\end{aligned} \tag{3-13}$$

对应的数值下界和上界分别为

$$\begin{aligned}l_{3,1,2} &= \gamma_{\geq}^{\star}\left(0.5\left(-x_{1,1}+x_{1,2}\right)\right)\\ &= -0.5u_{1,1}+0.5l_{1,2}\\ &= -1\\ u_{3,1,2} &= \gamma_{\leq}^{\star}\left(0.5\left(-x_{1,1}+x_{1,2}\right)+1\right)\\ &= -0.5l_{1,1}+0.5u_{1,2}+1\\ &= 2.\end{aligned} \tag{3-14}$$

多路径反向界限传播取 $x_{3,1}$ 最终的数值上下界为

$$[l_{3,1}, u_{3,1}] = [l_{3,1,1}, u_{3,1,1}] \cap [l_{3,1,2}, u_{3,1,2}] = [0, 2]. \tag{3-15}$$

按照同样的做法，可得到 $x_{4,1}$ 的数值上下界为 $[l_{4,1}, u_{4,1}] = [-2, 2]$。因此 $x_{4,1}$ 是激活状态不确定的节点，其对应的抽象变换为

$$\begin{aligned}\mathrm{ReLU}_1^{\#}(a_{4,1,1}) &= a_{5,1,1} = \left(0, 0.5x_{4,1}+1\right)\\ \mathrm{ReLU}_2^{\#}(a_{4,1,2}) &= a_{5,1,2} = \left(0.5x_{4,1}, 0.5x_{4,1}+1\right).\end{aligned} \tag{3-16}$$

由 $a_{5,1,1}$ 和 $a_{5,1,2}$ 可按照上述方式得到 $x_{5,1}$ 的数值上下界为 $[0,2]$。在节点 $x_{6,1}$ 处，使用两条路径已经能够得到比 DeepPoly 更为精确的结果，因为有

$$\begin{aligned}l_{6,1,1} &= \gamma_{\geq}^{\star}\left(0.25x_{1,1}-0.25x_{1,2}-1.5\right) = -2\\ u_{6,1,1} &= \gamma_{\leq}^{\star}\left(-0.25x_{1,1}+0.25x_{1,2}+1.5\right) = 2\\ l_{6,1,2} &= \gamma_{\geq}^{\star}\left(-1\right) = -1\\ u_{6,1,2} &= \gamma_{\leq}^{\star}\left(1\right) = 1.\end{aligned} \tag{3-17}$$

多路径反向界限传播方法取 $[l_{6,1}, u_{6,1}] = [-2,2] \cap [-1,1] = [-1,1]$，它包含于 DeepPoly 得到的数值上下界。更精确的数值上下界是界限传播方法的首要目标，因为它还可以用来构造更精确的近似。如图 3-2 给出单个 ReLU 节点的上近似，左图中 $x_{i,j}$ 的数值上下界 $[l,u]$ 给出的 ReLU 上近似为灰色区域。右图中更精确的数值上下界能够得到更精确的 ReLU 上近似（红色区域）。注意多路径界限传播方法区别于单纯使用多种界限传播方法计算输出层节点的多个可达集，因为它在每个节点均得到更精确的上下界，使得多条传播路径均可借以构造更精确的上近似，从而帮助后续计算积累精度优势，直到输出层。

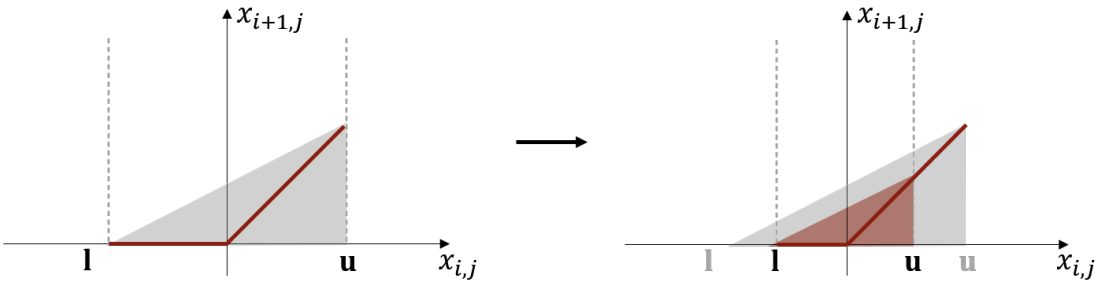


图 3-2 更准确的上下界带来更小的近似误差

3.3 多路径反向界限传播的可靠性证明

下面证明上述方法的可靠性，即任意节点 $x_{i,j}$ 的任意反向界限传播路径所对应的数值上下界 $\gamma^{\star}(a_{i,j,m})$，都是此节点在 ReLU 网络中数值上下界的上近似，这保证得到的可达集是 ReLU 网络的上近似。

注意到 $a_{i,j,m}$ 来自对 ReLU 以及仿射变换函数的抽象变换，对于激活状态确定的

ReLU 以及仿射变换函数，其抽象变换是准确的。这里只对激活状态不确定的 ReLU 节点的抽象变换 $\mathrm{ReLU}_m^{\#}$ 证明可靠性，其他抽象变换同理。因此可假设激活状态确定的 ReLU 以及仿射变换抽象已经具有可靠性。

定理 2 若第 i 层为神经网络中的一个 ReLU 层，对于任意 $j\in[d_i]$ 和任意 $m\in[M]$，上述方法得到的数值上下界

$$\left[l_{i+1,j,m},u_{i+1,j,m}\right]=\left[\gamma_{\geq}^{\star}\left(\gamma^{\geq}\left(\mathrm{ReLU}_m^{\#}(a_{i,j,m}^{\geq})\right)\right),\gamma_{\leq}^{\star}\left(\gamma^{\leq}\left(\mathrm{ReLU}_m^{\#}(a_{i,j,m}^{\leq})\right)\right)\right] \tag{3-18}$$

关于 ReLU 网络中节点 $x_{i+1,j}$ 的数值上下界可靠。

证明：归纳法。初始情况 $i=2$ 时，显然成立，因 $i=1$ 时仿射变换的抽象是可靠的（且此时没有误差），$i=2$ 时的可靠性由定理 1 保证。假设 $i=k-2$ 时上式成立，其中 k 为大于等于 4 的偶数。即 $x_{k-1,j}$ 的数值上下界 $[l_{k-1,j},u_{k-1,j}]=\cap_{m=1}^{M}[l_{k-1,j,m},u_{k-1,j,m}]$ 是可靠的。则由仿射变换抽象的可靠性可知 $[l_{k,j},u_{k,j}]=\cap_{m=1}^{M}[l_{k,j,m},u_{,j,m}]$ 关于 ReLU 网络中节点 $x_{k,j}$ 的数值上下界可靠，即 $i=k-1$ 时可靠。下面证明 $i=k$ 时可靠。因

$$\begin{aligned}\gamma\left(\mathrm{ReLU}_m^{\#}(a_{k,j,m})\right)&=\left(\gamma^{\geq}(a_{k+1,j,m}^{\geq}),\gamma^{\leq}(a_{k+1,j,m}^{\leq})\right)\\&=\left(\gamma^{\geq}(\lambda_m x_{k,j}),\gamma^{\leq}\left(\frac{u_{k,j}}{u_{k,j}-l_{k,j}}(x_{k,j}-l_{k,j})\right)\right)\\&=\left(\lambda_m\gamma^{\geq}\left(a_{k,j,m}^{\geq}\right),\frac{u_{k,j}}{u_{k,j}-l_{k,j}}\gamma^{\leq}\left(a_{k,j,m}^{\leq}\right)-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right).\end{aligned} \tag{3-19}$$

则有

$$\begin{aligned}\left[l_{k+1,j,m},u_{k+1,j,m}\right]&=\left[\gamma_{\geq}^{\star}\left(\lambda_m\gamma^{\geq}\left(a_{k,j,m}^{\geq}\right)\right),\gamma_{\leq}^{\star}\left(\frac{u_{k,j}}{u_{k,j}-l_{k,j}}\gamma^{\leq}\left(a_{k,j,m}^{\leq}\right)-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right)\right]\\&=\left[\lambda_m\gamma_{\geq}^{\star}\left(\gamma^{\geq}\left(a_{k,j,m}^{\geq}\right)\right),\frac{u_{k,j}}{u_{k,j}-l_{k,j}}\gamma_{\leq}^{\star}\left(\gamma^{\leq}\left(a_{k,j,m}^{\leq}\right)\right)-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right]\\&=\left[\lambda_m l_{k,j,m},\frac{u_{k,j}}{u_{k,j}-l_{k,j}}u_{k,j,m}-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right]\\&\supseteq\left[\lambda_m l_{k,j},\frac{u_{k,j}}{u_{k,j}-l_{k,j}}u_{k,j}-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right].\end{aligned} \tag{3-20}$$

由定理 1 可知，若 $[l_{k,j},u_{k,j}]$ 关于 ReLU 网络中 $x_{k,j}$ 可靠，则最后一行关于 ReLU 网络中 $x_{i+1,j}$ 可靠。因 $[l_{k,j},u_{k,j}]=\cap_{m=1}^{M}[l_{k,j,m},u_{,j,m}]$ 的可靠性（即当 $i=k-1$ 时）已知，故定理 2 成立。证毕。

直观而言，定理 2 说明对于任意激活状态不确定的节点 $x_{i,j}$，通过其 M 组符号约束反向传播到输入层得到的数值上下界关于 ReLU 网络都是可靠的，于是本文的方法得到每个节点真正可达集的上近似。

定理 3 多路径反向界限传播方法在每个节点得到的数值上下界包含于其任一条路径得到的数值上下界。即多路径反向界限传播方法的验证精度优于或等于任一条传播路径。

证明：设 $[l_{i,j},u_{i,j}]=\cap_{m=1}^{M}[l_{i,j,m},u_{i,j,m}]$ 为 M 条路径界限传播方法得到节点 $x_{i,j}$ 的数值上下界，若能证明 $[l_{i,j,m},u_{i,j,m}]\subseteq[l'_{i,j,m},u'_{i,j,m}]$ 对任意 m 成立，其中 $[l'_{i,j,m},u'_{i,j,m}]$ 为第 m 条路径得到的数值上下界，自然有 $[l_{i,j},u_{i,j}]\subseteq[l'_{i,j,m},u'_{i,j,m}]$。

第二归纳法。初始情况 $i=1$ 时以及 $i=2$ 时显然成立，因为此时分别对应输入层和仿射变换层，它们是准确的，且 $[l'_{i,j,m},u'_{i,j,m}]=[l_{i,j,m},u_{i,j,m}]$ 对任意 m 成立。假设对于任意 $i<k-1$ 成立 $[l_{i,j,m},u_{i,j,m}]\subseteq[l'_{i,j,m},u'_{i,j,m}]$，且第 $k-1$ 层为仿射变换层，下面证明 $i=k$ 时（此时为 ReLU 层）成立。

若有 $[l_{k,j,m},u_{k,j,m}]\subseteq[l'_{k,j,m},u'_{k,j,m}]$，蕴含对于任意 $\mathbf{x}_k$ 和 m，下式成立

$$\left[a^{\geq}_{k,j,m},a^{\leq}_{k,j,m}\right]\subseteq\left[a'^{\geq}_{k,j,m},a'^{\leq}_{k,j,m}\right] \tag{3-21}$$

其中 a' 为单独使用第 m 条传播路径对应的抽象元素。由式（3-20）可知

$$\begin{aligned}\left[l_{k,j,m},u_{k,j,m}\right]&=\left[\lambda_m\gamma^{\star}_{\geq}\left(\gamma^{\geq}\left(a^{\geq}_{k,j,m}\right)\right),\frac{u_{k,j}}{u_{k,j}-l_{k,j}}\gamma^{\star}_{\leq}\left(\gamma^{\leq}\left(a^{\leq}_{k,j,m}\right)\right)-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right]\\&\subseteq\left[\lambda_m\gamma^{\star}_{\geq}\left(\gamma^{\geq}\left(a'^{\geq}_{k,j,m}\right)\right),\frac{u_{k,j}}{u_{k,j}-l_{k,j}}\gamma^{\star}_{\leq}\left(\gamma^{\leq}\left(a'^{\leq}_{k,j,m}\right)\right)-\frac{u_{k,j}}{u_{k,j}-l_{k,j}}l_{k,j}\right]\\&=[l'_{k,j,m},u'_{k,j,m}]\end{aligned} \tag{3-22}$$

故 $[l_{k,j},u_{k,j}]\subset[l'_{k,j,m},u'_{k,j,m}]$ 成立。证毕。

定理 2 和定理 3 保证多路径反向界限传播方法能够得到更为精确且可靠的验证结果。

3.4 一维输入下的多路径反向界限传播方法

已有的反向界限传播方法（包括第 4 章介绍的其他界限传播方法）在一维输入下有更为直观的解释，它们为每个节点 $x_{i,j}$ 在给定输入范围内计算一组一维上下界函数。

若使用两条传播路径，则可得到每个节点 $x_{i,j}$ 的两组一维上下界函数，如图 3-3 所示。

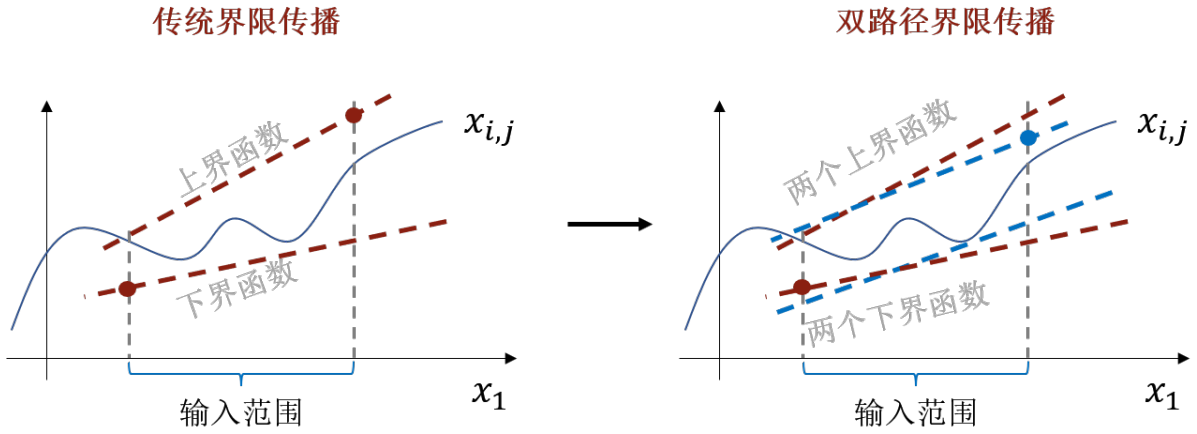


图 3-3 传统界限传播到双路径界限传播

在图 3-3 的右图中，双路径反向界限传播为每个节点 $x_{i,j}$ 在给定输入范围内计算红色和蓝色两组上下界函数（虚线），并选择它们中最好的一组（右图圆点）。这使得可以为每个节点得到更准确的数值上下界。如第 3.2 节所示，更准确的数值上下界意味着更小的近似误差，使得多条传播路径均可借以在每个节点构造更精确的上近似，从而帮助后续计算积累精度优势，直到输出层。

3.5 算法流程和复杂度

算法 1 给出了上述方法的伪代码，算法输入为网络 f，输入范围 $\mathcal{X}_1$ 以及对应的不安全区域 $\overline{S}$，算法判断网络 f 关于 $(\mathcal{X}_1,\overline{S})$ 是否安全。因为本文把仿射变换和 ReLU 变换作为分离的两层，所以需要根据不同的层类型来表示 $x_{i,j}$。在第 3 行，如果第 i 层为仿射变换层，那么需要用 $i-1$ 层节点仿射变换表示 $x_{i,j}$，这里 $\mathbf{w}_{i,j}$ 和 $b_{i,j}$ 分别表示节点 $x_{i,j}$ 的权重向量和偏移，$\mathbf{x}_{i-1}$ 表示 $i-1$ 层所有节点构成的向量。仿射变换的抽象表示是准确的，本文把这个准确的约束作为其抽象 $a_{i,j,m}$（3-9 式）。利用每个 $a_{i,j,m}$ 反向传播可得到 M 个数值上下界 $[l_{i,j,m},u_{i,j,m}]$，取它们的交集作为 $x_{i,j}$ 最终的数值上下界，如第 4~6 行所示。当第 i 层节点的数值上下界计算完成，就可以得到本层节点的可达集 $\mathcal{X}_i$，如 12 行所示。对于仿射变换层节点，其更精确的可达集在第 9 行被用来构造更精确的 M 种 ReLU 近似。按照相同的方法在 11 行得到 ReLU 层节点的数值上下界。第 13 行得到输出层节点可达集后，就可以判断网络 f 可达集与不安全区域 $\overline{S}$ 是否有交集。如果没

有交集返回 safe，说明 f 关于 $(\mathcal{X}_1,\overline{S})$ 安全；否则返回 unknown，因为此时不能保证 f 真正的可达集与 $\overline{S}$ 有交集。

算法 1 M 条路径反向界限传播算法

输入：网络 f，输入范围 $\mathcal{X}_1$ 和不安全区域 $\overline{S}$；
输出：如果安全返回 safe，否则返回 unknown。

1. **for** $i \leftarrow 1$ to L **do** // L 为网络 f 的层数
2. **for** $j \leftarrow 1$ to d_i **do** // d_i 为第 i 层节点数
3. **if** layer i is affine layer **then** // 如果第 i 层为仿射变换层
4. **for** $m \leftarrow 1$ to M **do**
5. $\left[l_{i,j,m},u_{i,j,m}\right] \leftarrow \left[\gamma_{\geq}^{\star}\left(\gamma^{\geq}(a_{i,j,m}^{\geq})\right),\gamma_{\leq}^{\star}\left(\gamma^{\leq}(a_{i,j,m}^{\leq})\right)\right]$
6. $[l_{i,j},u_{i,j}] \leftarrow \cap_{m=1}^{M}[l_{i,j,m},u_{i,j,m}]$ // 取 M 个数值上下界的交集
7. **if** layer i is ReLU layer **then** // 如果第 i 层为 ReLU 层
8. **for** $m \leftarrow 1$ to M **do** // 对于 $x_{i,j}$，用 $x_{i-1,j}$ 构造 M 种 ReLU 近似
9. $a_{i,j,m} \leftarrow \mathrm{ReLU}_m^{\#}(a_{i-1,j,m})$
10. $\left[l_{i,j,m},u_{i,j,m}\right] \leftarrow \left[\gamma_{\geq}^{\star}\left(\gamma^{\geq}(a_{i,j,m}^{\geq})\right),\gamma_{\leq}^{\star}\left(\gamma^{\leq}(a_{i,j,m}^{\leq})\right)\right]$
11. $[l_{i,j},u_{i,j}] \leftarrow \cap_{m=1}^{M}[l_{i,j,m},u_{i,j,m}]$
12. $\mathcal{X}_i \leftarrow [\mathbf{l}_i,\mathbf{u}_i]$ // 得到 $\mathcal{X}_i$ 为第 i 层节点的可达集
13. **if** $\mathcal{X}_L \cap \overline{S} = \varnothing$ **then** // 若输出层可达集与不安全区域交集为空
14. **return** safe
15. **else**
16. **return** unknown

算法 1 为上近似算法，返回 unknown 时，说明它得到的可达集 $\mathcal{X}_L$ 与 $\overline{S}$ 有交集，但 $\mathcal{X}_L$ 是放大的可达集。虽然无法确定真正的可达集是否与 $\overline{S}$ 存在交集，然而可以利用求解过程中得到的信息进一步求得更为精确的可达集，这一过程称为精化，文献[31,32]采用这种想法，以得到更为准确的验证结果。

算法 1 继承了基于简单线性近似的界限传播方法的高效性，其时间复杂度为$\mathcal{O}(MN^2)$，其中N为网络f的节点数量。对于每个节点，反向传播计算其数值上下界的时间代价是$\mathcal{O}(N)$。传播路径的数量M一般取低值常数，本文将在第 3.6.2 节讨论关于M的取值。算法 1 的空间复杂度是$\mathcal{O}(MN)$，对每个节点保存常数项个约束。

理论上路径的数量可以任意多，即M值可以任意大。那么一个值得关心的问题是，任意多的反向传播路径能够达到多好的精度？本文在第 3.6.2 节评估了随M值增加，验证精度的改善效果。

3.6 实验评估

同绝大部分工作一样，本文关注对一个具体输入的无穷范数扰动，并关注给定网络在无穷范数扰动下的鲁棒性。首先无穷范数扰动由如下的无穷范数距离定义。

定义 2（无穷范数距离）给定两个n维向量$\mathbf{x}=(a_i)$，$\mathbf{z}=(b_i)$，其中$1\le i\le n$，它们的无穷范数距离为$d_\infty(\mathbf{x},\mathbf{z})=\max\limits_{1\le i\le n}|a_i-b_i|$。

通常输入范围$\mathcal{X}_1$由上述度量给出，即给定具体输入$\mathbf{x}$和扰动大小θ，$\mathcal{X}_1=\mathcal{B}_\infty(\mathbf{x},\theta)\triangleq\{\mathbf{z}\,|\,d_\infty(\mathbf{x},\mathbf{z})\le\theta\}$。直观而言$\mathcal{X}_1$是$\mathbf{x}$每个维度最大允许加或减$\theta$所构成的向量集。

对于分类网络的验证问题，鲁棒半径能够很好地量化比较不同上近似方法的精度。下面是无穷范数距离下的鲁棒半径的定义，在后文中简称为鲁棒半径。

定义 3（d_∞鲁棒半径）对于分类网络f，给定输入$\mathbf{x}$及对应标签$label(\mathbf{x})$，f关于$\mathbf{x}$的d_∞鲁棒半径被定义为

$$R_f(\mathbf{x})=\max\left\{\left\{\theta\ge 0:f(\mathbf{z})=label(\mathbf{x}),\forall\mathbf{z}\in\mathcal{B}_\infty(\mathbf{x},\theta)\right\}\cup\{0\}\right\}. \tag{3-23}$$

对于分类网络，上述定义中$f(\mathbf{z})=label(\mathbf{x})$蕴含对不安全区域$\overline{S}$的选择是$\mathbf{x}$正确标签以外的所有标签，即针对无目标攻击的鲁棒性。直观而言，网络f关于输入$\mathbf{x}$的鲁棒半径，就是在不产生对抗样本的前提下，能对$\mathbf{x}$施加的最大扰动θ。

若f是归一化输入的网络，则有鲁棒半径$R_f(\mathbf{x})\in[0,1]$。通常使用二分法计算$R_f(\mathbf{x})$的值：取$\theta_1=1/2$作为初始值，在第i次迭代中，确定网络f关于扰动θ_i后的输入范围是否鲁棒。如果不鲁棒，将$\theta_{i+1}=\theta_i-(1/2)^{i+1}$作为下一个取值；如果鲁棒，则将

$\theta_{i+1}=\theta_i+(1/2)^{i+1}$作为下一个取值。重复上述迭代过程直至找到满足精度要求的θ_i，将其作为鲁棒半径$R_f(\mathbf{x})$的值。

给定神经网络和输入，上近似方法计算得到的鲁棒半径一般小于真正的鲁棒半径，这是因为上近似产生的误差通常会放大求得的可达集。因此，可以使用不同方法计算得到的鲁棒半径作为定量比较这些方法的精确度的指标。如果某种上近似方法得到的鲁棒半径更大，即更接近真正的鲁棒半径，则说明该上近似方法更精确。

本章将提出的多路径反向界限传播方法实现为 AbstraCMP[33]工具，其命名来源于该方法在神经网络各节点处对若干种近似的反向界限传播结果进行比较（CMP）。AbstraCMP 使用 Python 3.9 实现，其源代码可以在 https://github.com/formes20 获取。本节将在数据集 ACAS Xu[34]， MNIST[35]和 CIFAR-10[36]上将多路径反向界限传播方法与使用单条路径的代表性方法 DeepPoly 进行定量比较；在数据集 MNIST 上将多路径反向界限传播方法与单个节点线性近似最精确的方法 LP-ALL 进行定量比较。下面介绍实验使用的数据集和环境：

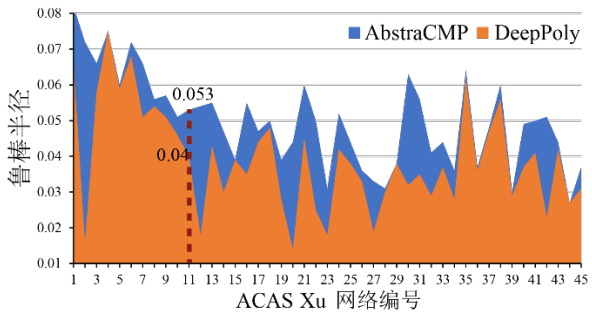
- ACAS Xu[34]网络用于小型无人机设备的防撞规避系统，由 45 个全连接前馈 ReLU 网络组成。这 45 个网络规模相同，均为6×50，其中 6 表示隐藏层层数，50 表示每个隐藏层节点个数。每个 ACAS Xu 网络输出层是 5 维，分别表示网络给出的 5 种碰撞规避决策。输入层是 5 维，分别表示与其他飞行设备的距离及角度等 5 个参数，是低维输入网络的代表。
- MNIST[35]是包含 0~9 十个手写数字图像的数据集，包含 6 万个训练样本以及 1 万个测试样本。每个样本都是灰阶图像且大小均为 28×28 像素。本文使用 MNIST 训练数据集训练了规模分别为 16×50，10×80 及 20×50 的三个全连接前馈 ReLU 网络，每个网络的输出层是 10 维，分别表示网络给出的 10 个分类结果。输入层为 28×28=784 维，表示一张图片的 784 个灰度值，是较高维输入网络的代表。
- CIFAR-10[36]是包含 10 种物体图像的数据集，共计 5 万个训练样本和 1 万个测试样本。每个样本为 3 通道 RGB 图像且大小均为 32×32 像素。本文使用 CIFAR-10 训练数据集训练了规模分别为 10×100，15×200 及 16×250 的三个全连接前馈 ReLU 网络，每个网络的输出层是 10 维，分别表示网络给出的 10 种分类结果。输入层为 3×32×32=3072 维，表示一张图片的 3072 个灰度值，是高维输入网络的代表。

上述网络均使用归一化输入训练，因此关于某个给定输入的扰动是归一化后的扰动，关于给定输入的鲁棒半径也是归一化后的鲁棒半径。

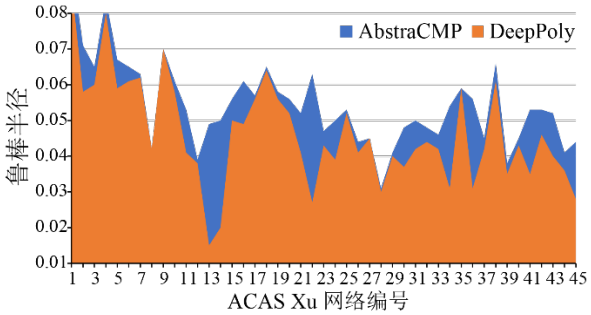
本章使用的实验平台参数为：处理器 Intel Core i7-6700 且主频 3.40GHz；内存大小 16GB；操作系统版本 Windows 10 专业版 64 位；Python 版本 3.9。与工具 DeepPoly 和 LP-ALL 的对比实验均在相同环境下进行。

3.6.1 低维输入网络上的精度改善

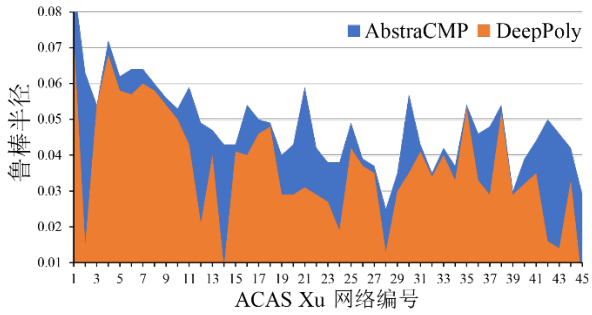
对于 ACAS Xu 数据集网络，为了方便对比，本文选择 4 个随机合法输入，分别通过 AbstraCMP 和 DeepPoly 计算这 4 个输入在 45 个网络上的鲁棒半径。本实验中设定鲁棒半径精确到千分位；AbstraCMP 使用 $M=3$，三条界限传播路径分别为 DeepPoly 路径，直角三角形近似和钝角三角形近似对应的反向界限传播路径。结果如图 3-4 所示。



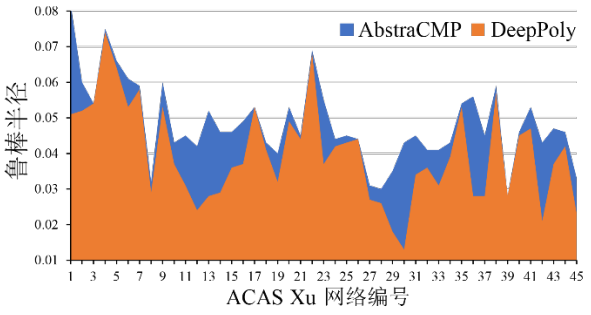
(a) 输入 1 在 45 个网络上的鲁棒半径



(b) 输入 2 在 45 个网络上的鲁棒半径



(c) 输入 3 在 45 个网络上的鲁棒半径



(d) 输入 4 在 45 个网络上的鲁棒半径

图 3-4 ACAS Xu 网络上能够验证的鲁棒半径对比

图 3-4（a）展示了对于输入 1，工具 AbstraCMP 和 DeepPoly 在 45 个网络中能验证的鲁棒半径。其中深蓝区域的上边界为 AbstraCMP 能够验证的鲁棒半径，浅橙色为 DeepPoly 能够验证的鲁棒半径。例如，图 3-4（a）中虚线表示，在第 11 个网络上，两

种方法得到的鲁棒半径分别为 0.053 和 0.04，提高了（0.053－0.04）/ 0.04 = 32.5%。对于任意输入和任意网络，AbstraCMP 在理论上保证至少达到 DeepPoly 的效果，因此 AbstraCMP 可验证的鲁棒半径（深蓝色区域）总是大于 DeepPoly 可验证的鲁棒半径（浅橙色区域）。通过图 3-4（b）（c）（d） 均能观察到同样的结论。对于上近似方法，同一输入下能验证更大的鲁棒半径，意味着对真正可达集更精确的近似。本实验中 AbstraCMP 能够验证的单个鲁棒半径最大比 DeepPoly 提高 370%；对于每个输入，在 45 个网络上能够验证的鲁棒半径平均值比 DeepPoly 提高 25~30%。实验结果表明，工具 AbstraCMP 在低维输入网络中的验证精度相比 DeepPoly 有所提升。

3.6.2 反向界限传播路径数量对精度和时间的影响

理论上反向界限传播路径可以任意多。通过下面的实验，本文将研究反向界限传播路径数量的增加对鲁棒半径计算的改进以及带来的时间代价。固定一个 ACAS Xu 网络，并选定一个输入，本实验评估使用不同数量的路径时，AbstraCMP 所能验证的鲁棒半径大小以及对应的时间开销。

关于路径的选择，本实验首先选取 DeepPoly 路径作为 $M=1$ 的配置，此外由于定理 1 中任意 $\lambda \in [0,1]$ 都构成对 ReLU 激活函数的可靠近似，所以可选取 $[0,1]$ 区间的等分值构成若干条路径。具体而言，$M=2$ 时包含 DeepPoly 和 $\lambda=0$ 两条路径；$M=3$ 时包含 DeepPoly 和 $\lambda=0$ 及 $\lambda=1$ 共 3 条路径；$M=4$ 时包含 DeepPoly 及 $\lambda=0, 0.5, 1$ 共 4 条路径；以此类推。本实验中设定鲁棒半径精确到千分位，得到 AbstraCMP 能够验证的鲁棒半径随 M 值的变化如图 3-5 所示。

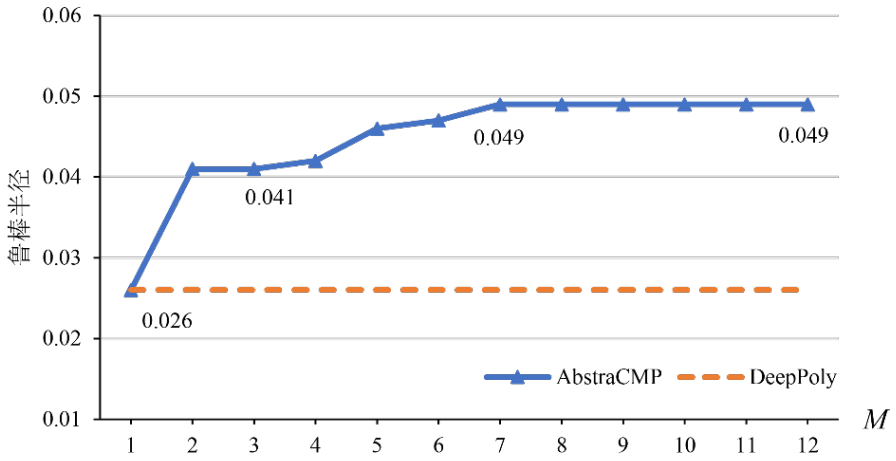


图 3-5 AbstraCMP 可验证鲁棒半径随 M 值的变化关系

图 3-5 中带有三角形标签的蓝色实折线表示在给定网络和输入下，随路径数量 M 增加，AbstraCMP 能够验证的鲁棒半径变化。橙色虚线表示 DeepPoly 在该网络和输入下能够验证的鲁棒半径，虽然 DeepPoly 没有路径的概念，但为了方便对比，图中将其能够验证的鲁棒半径 0.026 作为基准线。从图 3-5 可以看出，在 DeepPoly 的基础上，增加较少的路径就能取得相对于 DeepPoly 较好的效果。在本实验的网络和输入下，仅使用 $M=2$ 即可使计算得到的鲁棒半径相对于 DeepPoly 提高 57%。

M 值的增加可以获得更为精确的结果，但这种改善并不能持续。在该实验中，随 M 值增加，AbstraCMP 能够验证的鲁棒半径会迅速收敛。如图 3-5 所示，当 $M=7$ 时就已经达到收敛值 0.049。其它实验也表明，对于大部分输入，通常 $M=3$ 或 $M=4$ 已经接近收敛值。因此为了平衡精度与时间，$M=3$ 是一个较为合理的选择。

表 3-6 AbstraCMP 验证时间随 M 值的变化关系

方法	DeepPoly	AbstraCMP								
		$M=2$	$M=3$	$M=4$	$M=5$	$M=6$	$M=7$	$M=8$	$M=9$	$M=10$
用时（秒）	6.44	12.60	20.79	28.02	35.80	42.93	51.01	58.07	65.38	72.86

表 3-6 给出了在上述网络和输入下，AbstraCMP 计算对应鲁棒半径所需时间随反向界限传播路径数量 M 的变化关系。表中第二列是 DeepPoly 的用时，第三列 $M=2$ 及之后各列是取不同 M 值时 AbstraCMP 的用时。对于每一列的配置，实验记录其运行 10 次的平均时间。正如第 3.5 节的讨论，AbstraCMP 的时间代价与路径数量 M 线性相关，从表 3-6 中也可以看出这一关系。DeepPoly 解决神经网络验证问题的时间代价是极低的，当 M 取低值常数时，AbstraCMP 相较于现有验证方法而言时间代价也是极低的。因此本文的方法所引入的额外时间代价很小。

值得注意的是，本文的方法在效率上存在较大的提升空间。AbstraCMP 目前虽然使用简单的单线程实现，但多路径反向界限传播方法在理论上可以高度并行化。对于单条路径，同层各节点的数值上下界计算可以并行进行。而对于 AbstraCMP，除同层各节点的并行以外，同一节点不同传播路径对应的数值上下界也可以并行计算，以充分利用 CPU 甚至 GPU 的计算能力。本文在第 4 章实现这一技术。

3.6.3 高维输入网络上的精度改善

本节在数据集 MNIST 和 CIFAR-10 上测试 AbstraCMP 在高维输入网络上的表现。对于 MNIST 数据集，本实验使用其训练集训练了三个全连接前馈神经网络，它们的隐藏层的规模分别为 16×50，10×80 和 20×50。实验中选择测试集的前 100 张图片作为输入，对比 AbstraCMP 和 DeepPoly 在不同扰动大小θ下能够成功验证鲁棒性的图片数量。对于上近似方法，给定一组输入和具体的θ值，能够验证的鲁棒性问题越多，说明这种方法越精确。本实验中 AbstraCMP 取$M=3$，结果如表 3-7 所示。

表 3-7 不同扰动下 DeepPoly 和 AbstraCMP 成功验证鲁棒性的图片数量（MNIST）

网络	方法	扰动大小θ								
		0.010	0.012	0.015	0.017	0.020	0.022	0.025	0.027	0.030
MNIST 16×50	DeepPoly	89	82	71	66	52	40	26	23	16
	AbstraCMP	**90**	**86**	**76**	**70**	**58**	**45**	**28**	**23**	**18**
MNIST 10×80	DeepPoly	91	88	79	67	46	33	27	20	10
	AbstraCMP	**94**	**91**	**82**	**70**	**48**	**38**	**31**	**24**	**12**
MNIST 20×50	DeepPoly	73	70	49	40	31	22	16	13	7
	AbstraCMP	**80**	**72**	**58**	**49**	**37**	**27**	**20**	**18**	**10**

从表中可以看到，对于同一网络和相同扰动大小，AbstraCMP 能够验证的图片数量总是大于 DeepPoly。例如，在网络 MNIST 20×50 上，选定扰动大小θ为 0.010 时，DeepPoly 能够验证鲁棒的图片数量是 73，而 AbstraCMP 能够验证 80 张图片的鲁棒性。实验结果表明，AbstraCMP 在较高维输入网络上的计算精度相比 DeepPoly 同样有所提升。

在本实验中，对相同扰动大小，AbstraCMP 能够成功验证的图片数量最多比 DeepPoly 多 42.9%（=（10−7）/ 7），此情况出现在 MNIST 20×50 网络且扰动大小$\theta=0.030$时。而且从实验结果可以观察到，当网络隐藏层的数量较多时，AbstraCMP 相比 DeepPoly 提升更加明显。产生这一结果的原因是，随着网络深度增加，多路径反向界限传播方法在每个节点相较于单路径界限传播提升的精度得到积累和放大。

对于 CIFAR-10 数据集，本实验使用其训练集训练了隐藏层的规模分别为 10×100，15×200 及 16×250 的三个全连接前馈神经网络。只有 ReLU 层的全连接

CIFAR-10 网络相对于 MNIST 网络有较低的准确率和较小的鲁棒半径，因此对于每个网络，实验选择测试集中分类正确的前 100 张图片作为输入，对比 AbstraCMP 和 DeepPoly 在不同扰动大小θ下能够成功验证鲁棒性的图片数量。本实验中 AbstraCMP 取$M=3$，结果如表 3-8 所示。

表 3-8 不同扰动下 DeepPoly 和 AbstraCMP 成功验证鲁棒性的图片数量（CIFAR-10）

网络	方法	扰动大小θ								
		0.0005	0.0010	0.0015	0.0020	0.0025	0.0030	0.0035	0.0040	0.0045
CIFAR-10 10×100	DeepPoly	95	89	75	62	48	42	30	24	17
	AbstraCMP	**95**	**91**	**81**	**67**	**60**	**50**	**40**	**35**	**23**
CIFAR-10 15×200	DeepPoly	93	87	75	64	57	48	35	32	21
	AbstraCMP	**94**	**88**	**79**	**70**	**61**	**55**	**47**	**35**	**29**
CIFAR-10 16×250	DeepPoly	93	76	59	42	33	20	14	8	3
	AbstraCMP	**95**	**79**	**62**	**49**	**37**	**23**	**16**	**10**	**4**

在本实验中，对相同扰动大小，AbstraCMP 最多能够比 DeepPoly 多验证 12 张图片的鲁棒性，此情况出现在扰动大小为 0.0035 及网络规模为 15×200 时。且在扰动大小为 0.0035 时，AbstraCMP 在三个网络上平均比 DeepPoly 多验证 8 张图片的鲁棒性。

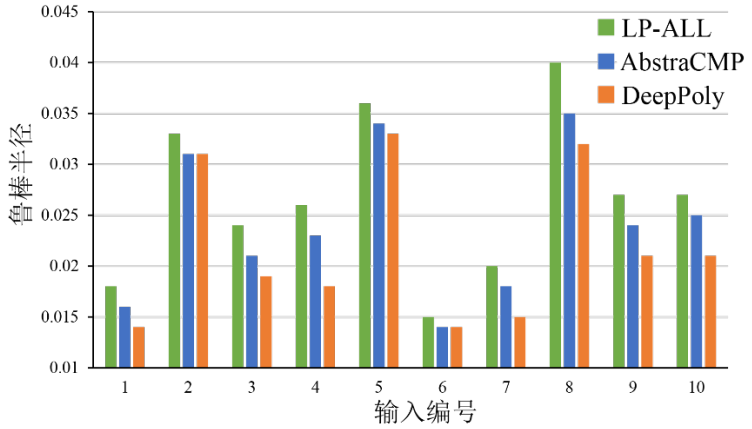
上述三组实验说明，在不同规模的输入和网络下，AbstraCMP 的验证精度相较于 DeepPoly 均有所提升。

3.6.4 与 LP-ALL 的精度比较

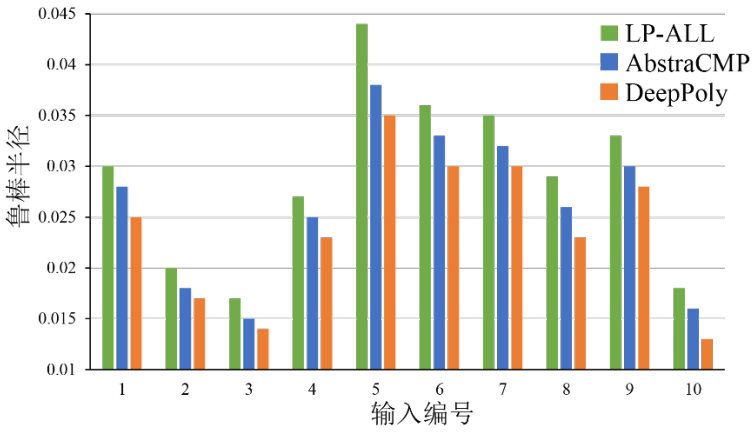
Salman 等人[22]将单个神经网络节点的线性近似方法统一为框架，并说明这类方法存在精度上限。对于 ReLU 网络，文中将其精度上限对应的方法称为 LP-ALL。LP-ALL 逐层逐节点构造 LP 问题以得到每个节点的数值上下界，并使用得到的数值上下界构造 ReLU 近似。这是目前 ReLU 网络单个节点的线性凸松弛方法能够得到的最精确上下界。另一方面，LP-ALL 需要解$\mathcal{O}(N)$个 LP 问题，其中N为给定网络的节点数目，且平均每个 LP 问题的变量数目为$\mathcal{O}(N)$个，这使得目前 LP-ALL 对于较大规模网络是不太现实的方法。Salman 等人的文章[22]使用 1000 CPU 节点的集群计算了 MNIST 全体测试集图片对于给定扰动大小的鲁棒比例，实验使用 1×500 和 2×100 两种规模的

MNIST 网络，总计算量已经超过 22 CPU 年。

本节将定量比较 LP-ALL，AbstraCMP 和 DeepPoly 的验证精度。具体地，实验选择第 3.6.3 节中的两个网络：MNIST 10×80 和 MNIST 16×50，每个网络分别选择 10 张图片作为输入，比较三种方法得到这 10 个输入的鲁棒半径。其中 AbstraCMP 使用路径数目 $M=3$，LP-ALL 求解器使用与文献相同的 ECOS，实验中设定鲁棒半径的精度阈值为千分位，得到结果如图 3-9 所示。



(a) MNIST 16×50 网络的鲁棒半径对比



(b) MNIST 10×80 网络的鲁棒半径对比

图 3-9 LP-ALL，AbstraCMP，DeepPoly 能够验证的鲁棒半径对比

从图 3-9 中可看出，正如 Salman 等人提到的，LP-ALL 作为一种复杂度很高的方法，却不能显著提升已有线性近似方法的验证精度。对于本实验使用的全体输入集，LP-ALL 得到一个千分位鲁棒半径平均耗时 148×10^3 秒（约 41 小时），但其精度相较 DeepPoly 仅平均绝对提升 0.0050；而 AbstraCMP 得到一个千分位鲁棒半径平均耗时 435 秒，其精度相较 DeepPoly 已具有 0.0023 的平均绝对提升。同时由图 3-9 也可知，

AbstraCMP 在精度上弥补了 DeepPoly 和 LP-ALL 的间隙，使得时间复杂度较低的界限传播方法能够更加接近 LP-ALL 的验证效果。

3.7 本章小结

针对反向界限传播方法，本章提出反向界限传播路径的概念，在这一概念下，现有的反向界限传播方法仅使用一条传播路径，是本章所提出的多路径反向界限传播方法的特例。本章形式化地说明了多路径反向界限传播方法的算法过程，并证明其可靠性，同时从理论上保证了其能够提升验证精度。

在实验部分，本章将多路径反向界限传播方法在 ACAS Xu，MNIST 和 CIFAR-10 三个常用数据集上与现有反向界限传播方法进行验证精度的比较，实验结果说明本方法有明显的验证精度提升。本章同时展示了反向界限传播路径数目对验证精度的影响；以及多路径反向界限传播方法与单个节点线性近似的最精确方法 LP-ALL 的验证精度比较。

第 4 章 基于 GPU 并行的多路径界限传播

本章将上一章反向界限传播路径的概念扩展到其他类型的界限传播，包括前向、前向+反向等，并扩展为它们对应的多路径界限传播方法。同时，利用多条界限传播路径的独立性，本章将界限传播计算过程在 GPU 上并行化，使得多路径界限传播方法的实际时间复杂度降低到传统的单条传播路径的水平。本章在机器学习框架 PyTorch 上重新实现并扩展了多路径界限传播验证方法，以提供高效的 GPU 并行验证。

本章内容组织如下：第 4.1 节将会介绍扩展的多路径前向界限传播方法；第 4.2 节将会介绍本章扩展实现的工具 MpBP，它集成了四种典型的界限传播方法；第 4.3 节将会给出 MpBP 的使用示例；第 4.4 节将 MpBP 与已有的最先进界限传播验证工具进行实验比较；第 4.5 节给出本章总结。

4.1 多路径前向界限传播

由第 3 章中的介绍，反向界限传播类似神经网络的训练过程，在实现上以二重嵌套循环的方式计算每个节点的界限函数。外部循环从输入层到输出层逐层遍历，对于每层的节点，内部循环通过反向替换的方式向输入层迭代计算其关于输入层节点的线性表示，因此“反向”一词来自内部循环。之后通过输入范围可将这两个界限函数具体化，得到节点的数值取值范围。这些范围又被用来计算下一层节点的界限函数。

前向界限传播仅从输入层到输出层逐层遍历，计算每层节点的界限函数，而不执行反向传播（即内层循环），它类似神经网络的推导过程，因此效率较高；另一方面，它得到的节点取值范围不如反向界限传播精确。与反向界限传播相比，前向界限传播为了效率牺牲了精确度，但在分支定界法等完备验证方法中，它可作为重要的定界方法。在分支定界法中，界限传播通常用于帮助决定下一个分支的节点。这种需求要求界限传播技术足够快速，并具有可接受的精确度，因此前向界限传播是一个较好的选择。与反向界限传播一样，现有的前向界限传播方法只使用单条传播路径。本章将传统的前向界限传播扩展到多路径前向界限传播，它同样能够在理论上保证比传统的单路径前向界限传播更精确。

4.1.1 多路径前向界限传播

图 4-1 举例说明了一个简单神经网络的两条路径前向界限传播。假设所有节点的偏移都为 0。图中输入层节点为$x_{1,1}$和$x_{1,2}$，前向界限传播从输入层出发，利用相邻层之间的符号约束前向地逐层计算每个节点的界限函数（而没有替换到输入层的步骤）。对于节点$x_{i+1,1}$，两条前向界限传播路径为其确定两个下界函数$low_{i+1,1,1}$和$low_{i+1,1,2}$，它们由$x_{i,1}$的界限函数直接通过 ReLU 线性近似变换得到。由界限函数$low_{i+1,1,1}$和$low_{i+1,1,2}$可得到节点$x_{i+1,1}$的两个数值下界$l_{i+1,1,1}$和$l_{i+1,1,2}$，取其最终的数值下界为$\max(l_{i+1,1,1}, l_{i+1,1,2})$可保证得到相较于单条路径更精确的下界。

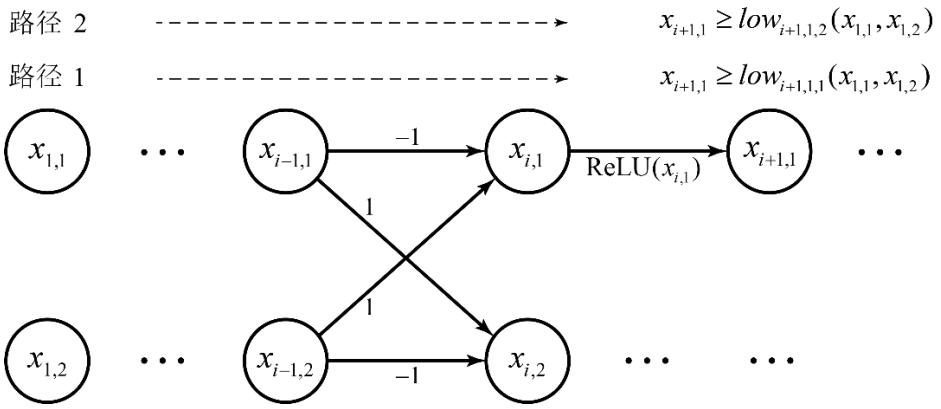


图 4-1 两条传播路径的前向界限传播

形式化地，若沿用第 3 章的记号，每个节点$x_{i+1,j}$关于上层节点的符号约束$a^{\geq}_{i,j,m}$和$a^{\leq}_{i,j,m}$表示为：

$$\sum_{k=1}^{d_i} q_{i+1,k,m} x_{i,k} + r_{i+1,j,m}, \tag{4-1}$$

则多路径的前向界限传播从第i层得到第$i+1$层的界限函数过程为：

$$\begin{aligned}
&low_{1,j,m} = up_{1,j,m} = x_{1,j} \\
&low_{i+1,j,m} = \sum_{k=1}^{d_i} \frac{|\delta(q_{i+1,k,m})+1|}{2} q_{i+1,k,m} low_{i,k,m} + \sum_{k=1}^{d_i} \frac{|\delta(q_{i+1,k,m})-1|}{2} q_{i+1,k,m} up_{i,k,m} + r_{i+1,j,m} \\
&up_{i+1,j,m} = \sum_{k=1}^{d_i} \frac{|\delta(q_{i+1,k,m})+1|}{2} q_{i+1,k,m} up_{i,k,m} + \sum_{k=1}^{d_i} \frac{|\delta(q_{i+1,k,m})-1|}{2} q_{i+1,k,m} low_{i,k,m} + r_{i+1,j,m}
\end{aligned} \tag{4-2}$$

上式中δ为符号函数，当$q\geq 0$时，$\delta(q)=+1$；$q<0$时，$\delta(q)=-1$，它用于根据括号内系数q的正负帮助得到下界函数和上界函数。例如，对于下界函数$low_{i+1,j,m}$，若$q_{i+1,k,m}>0$，使用$low_{i,k,m}$；若$q_{i+1,k,m}<0$，则使用$up_{i,k,m}$。系数$q_{i+1,k,m}$的具体值取决于当前层类型。例如，对于 ReLU 网络，相邻层之间有式（3-8）至（3-10）三种线性变换，故系数$q_{i+1,k,m}$由这三种线性变换逐层复合得到。

区别于第 3 章多路径反向界限传播，上式没有$\gamma^{\geq}(a^{\geq}_{i,k,m})$和$\gamma^{\leq}(a^{\leq}_{i,k,m})$定义的互递归函数递归到输入层，而仅从上层节点关于输入层$x_{1,k}$的界限函数得到下层节点关于输入层$x_{1,k}$的界限函数。注意由于界限函数$low_{i,k,m}$和$up_{i,k,m}$是关于输入层的函数而非前一层的函数，所以上式不能诠释为递归到输入层的过程。上式的可靠性由如下定理 4 保证。

定理 4 对于任意$i\in[L]$，任意$j\in[d_i]$和任意$m\in[M]$，多路径前向界限传播得到的数值上下界

$$\left[l_{i,j,m},u_{i,j,m}\right]=\left[\gamma^{\star}_{\geq}\left(low_{i,j,m}\right),\gamma^{\star}_{\leq}\left(up_{i,j,m}\right)\right] \tag{4-3}$$

关于 ReLU 网络中节点$x_{i+1,j}$的数值上下界可靠。

证明：归纳法。初始情况$i=1$以及$i=2$时，显然成立，因为分别对应输入层和仿射变换，而仿射变换的抽象是可靠的。假设$i=k-1$时上式成立，即$x_{k-1,j}$的数值上下界$[l_{k-1,j},u_{k-1,j}]=\cap_{m=1}^{M}[l_{k-1,j,m},u_{k-1,j,m}]$是可靠的。下面证明$i=k$时可靠。若$x_{k,j}$为激活状态不确定的 ReLU 节点，即有

$$a_{k,j,m}=\mathrm{ReLU}^{\#}_{m}(a_{k-1,j,m})=\left(\lambda_m x_{k-1,j},\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}x_{k-1,j}-\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}l_{k-1,j}\right). \tag{4-4}$$

则有

$$\begin{aligned}\left[l_{k,j,m},u_{k,j,m}\right]&=\left[\gamma^{\star}_{\geq}\left(low_{k,j,m}\right),\gamma^{\star}_{\leq}\left(up_{k,j,m}\right)\right]\\&=\left[\gamma^{\star}_{\geq}\left(\lambda_m\cdot low_{k-1,j,m}\right),\gamma^{\star}_{\leq}\left(\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}\cdot up_{k-1,j,m}-\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}l_{k-1,j}\right)\right]\\&=\left[\lambda_m\gamma^{\star}_{\geq}\left(low_{k-1,j,m}\right),\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}\gamma^{\star}_{\leq}\left(up_{k-1,j,m}\right)-\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}l_{k-1,j}\right]\\&=\left[\lambda_m\cdot l_{k-1,j,m},\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}u_{k-1,j,m}-\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}l_{k-1,j}\right].\end{aligned} \tag{4-5}$$

由定理 1 可知，若$[l_{k-1,j},u_{k-1,j}]$关于 ReLU 网络中$x_{k,j}$可靠，则最后一行$[l_{k,j,m},u_{k,j,m}]$关于

ReLU 网络中$x_{k,j}$可靠。因$[l_{k-1,j},u_{k-1,j}]=\cap_{m=1}^{M}[l_{k-1,j,m},u_{k-1,j,m}]$的可靠性（即当$i=k-1$时）已知，故$[l_{k,j,m},u_{k,j,m}]$可靠。对于仿射变换层节点和激活状态确定的 ReLU 节点同理可证，即定理 4 成立。证毕。

定理 5 多路径前向界限传播方法在每个节点得到的数值上下界包含于其任一条路径得到的数值上下界。即多路径前向界限传播方法的验证精度优于或等于任一条传播路径。

证明：设$[l_{i,j},u_{i,j}]=\cap_{m=1}^{M}[l_{i,j,m},u_{i,j,m}]$为$M$条路径界限传播方法得到节点$x_{i,j}$的数值上下界，若能证明$[l_{i,j,m},u_{i,j,m}]\subseteq[l'_{i,j,m},u'_{i,j,m}]$对任意$m$成立，其中$[l'_{i,j,m},u'_{i,j,m}]$为第$m$条路径得到的数值上下界，自然有$[l_{i,j},u_{i,j}]\subseteq[l'_{i,j,m},u'_{i,j,m}]$。

归纳法。初始情况$i=1$时以及$i=2$时显然成立，此时分别对应输入层和仿射变换层，它们是准确的，且$[l'_{i,j,m},u'_{i,j,m}]=[l_{i,j,m},u_{i,j,m}]$对任意$m$成立。假设$i=k-1$时成立$[l_{k-1,j,m},u_{k-1,j,m}]\subseteq[l'_{k-1,j,m},u'_{k-1,j,m}]$，且第$k-1$层为仿射变换层，下面证明$i=k$时（ReLU 层）此式成立。

若有$[l_{k-1,j,m},u_{k-1,j,m}]\subseteq[l'_{k-1,j,m},u'_{k-1,j,m}]$，蕴含对于任意$m$，下式成立

$$\left[\gamma_{\geq}^{\star}(low_{k-1,j,m}),\gamma_{\leq}^{\star}(up_{k-1,j,m})\right]\subseteq\left[\gamma_{\geq}^{\star}(low'_{k-1,j,m}),\gamma_{\leq}^{\star}(up'_{k-1,j,m})\right] \tag{4-6}$$

其中low'和up'为单独使用第m条传播路径对应的抽象元素。由式（4-5）可知

$$\begin{aligned}\left[l_{k,j,m},u_{k,j,m}\right]&=\left[\lambda_m\gamma_{\geq}^{\star}\left(low_{k-1,j,m}\right),\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}\gamma_{\leq}^{\star}\left(up_{k-1,j,m}\right)-\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}l_{k-1,j}\right]\\&\subseteq\left[\lambda_m\gamma_{\geq}^{\star}\left(low'_{k-1,j,m}\right),\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}\gamma_{\leq}^{\star}\left(up'_{k-1,j,m}\right)-\frac{u_{k-1,j}}{u_{k-1,j}-l_{k-1,j}}l_{k-1,j}\right]\\&=[l'_{k,j,m},u'_{k,j,m}]\end{aligned} \tag{4-7}$$

故$[l_{k,j},u_{k,j}]\subseteq[l'_{k,j,m},u'_{k,j,m}]$成立。

定理 4 和定理 5 保证多路径前向界限传播能够得到更为精确且可靠的验证结果。

4.1.2 算法流程和复杂度

算法 2 给出了多路径前向界限传播计算给定网络输出范围的伪代码。第 1~2 行初始化输入层节点的界限函数，第 3~5 行逐层地计算每个节点的M个界限函数，第 7~8 行得到每个节点的数值上下界，第 10 行返回输出层节点的取值范围$\mathcal{X}_L$。

算法 2 M 条路径前向界限传播计算给定网络的输出范围

输入：网络 f，输入范围 $\mathcal{X}_1$；
输出：网络 f 在输入范围 $\mathcal{X}_1$ 下计算得到的输出范围 $\mathcal{X}_L$

1.**for** $j \leftarrow 1$ to d_1 **do** // d_1 为输入层节点数
2. $low_{1,j,m} \leftarrow x_{1,j}, up_{1,j,m} \leftarrow x_{1,j}$ // 初始化界限函数
3. **for** $i \leftarrow 2$ to L **do** // L 为网络 f 的层数
4. **for** $j \leftarrow 1$ to d_i **do** // d_i 为第 i 层节点数
5. **for** $m \leftarrow 1$ to M **do**
6. 计算 $low_{i,j,m}$ 与 $up_{i,j,m}$ // 计算每个节点的界限函数
7. $\left[l_{i,j,m}, u_{i,j,m}\right] \leftarrow \left[\gamma_{\geq}^{\star}\left(low_{i,j,m}\right), \gamma_{\leq}^{\star}\left(up_{i,j,m}\right)\right]$
8. $[l_{i,j}, u_{i,j}] \leftarrow \cap_{m=1}^{M}[l_{i,j,m}, u_{i,j,m}]$ // 取 M 个数值上下界的交集
9. $\mathcal{X}_i \leftarrow [\mathbf{l}_i, \mathbf{u}_i]$ // 得到 $\mathcal{X}_i$ 为第 i 层节点的可达集
10. return $\mathcal{X}_L$

多路径前向界限传播类似神经网络的推理过程，仅对各节点一次遍历，其时间复杂度为 $\mathcal{O}(MN)$，其中 N 为网络节点数目。其空间复杂度为 $\mathcal{O}(MN)$，为每个节点保存常数个约束。

前向界限传播与反向界限传播可混合使用。例如，前向+反向界限传播[5]是前向界限传播与反向界限传播在验证精度上的一种折衷，它从输入层开始前向地逐层计算每个节点的界限，而仅在最后一层节点进行反向界限传播。它得到的界限函数比前向界限传播更为精确，弥补了前向界限传播和反向界限传播之间的精度和时间差距。本文将其扩展到多路径前向+反向界限传播，扩展后的方法也弥补了多路径前向界限传播与多路径反向界限传播之间的精度间隙。本文将在第 4.4 节中评估前向+反向界限传播的实际效果。

4.2 MpBP 工具

MpBP 是本文基于 PyTorch 实现的、能够利用 GPU 并行化多路径界限传播的验证工具。借助 PyTorch 提供的张量运算并行框架，多条界限传播路径可以作为高维张量的一个维度，从而在 GPU 上达到与传统界限传播方法相当的时间复杂度。MpBP 的

GitHub 开源页面 https://github.com/formes20 提供了更多介绍和使用说明。具体而言，MpBP 区别于第 3 章 AbstraCMP 的两个主要贡献是：

1. 它是一个多路径界限传播验证工具集。MpBP 支持四种界限传播方法：第 3 章介绍的多路径反向界限传播、第 4.1 节介绍的多路径前向界限传播、以及多路径前向+反向界限传播和传统的区间传播。它们在验证精度和时间消耗之间进行了各种权衡，可以适应不同的应用场景。
2. 它是一个基于 PyTorch 的并行验证工具。通过 PyTorch 提供的高效并行张量计算框架，多路径界限传播与传统的单条路径界限传播有几乎相同的时间开销。另一方面，对于已经熟悉 PyTorch 的用户，MpBP 提供了更轻松和更灵活的使用和扩展。

4.2.1 MpBP 框架

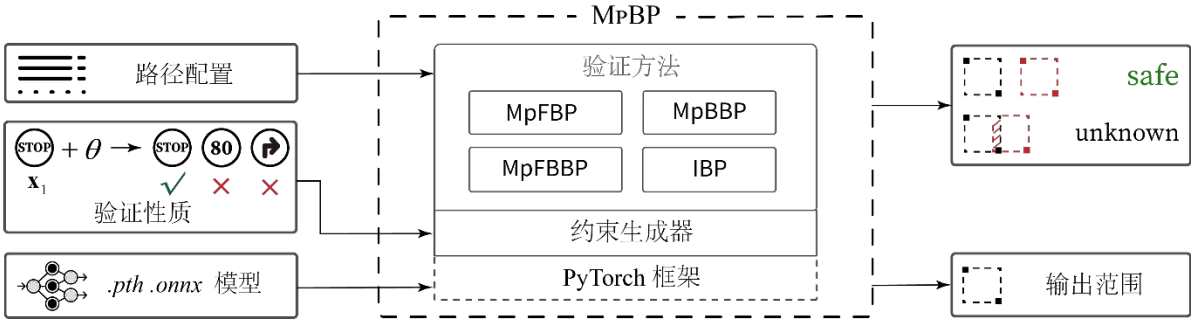


图 4-2 工具 MpBP 的框架

MpBP 的框架和工作流程如图 4-2 所示。MpBP 接受神经网络模型、验证性质和路径配置作为输入，输出给定模型是否满足预期性质，若满足则返回 safe，表示验证性质成立；否则返回 unknown，表示未知。MpBP 的核心由四个模块组成，分别是多路径前向界限传播（MpFBP）、多路径反向界限传播（MpBBP）、多路径前向+反向界限传播（MpFBBP）和区间传播（IBP），它们用来计算给定神经网络的输出范围。这些模块对验证精度和时间消耗进行了不同程度的权衡，用户可以根据自己的需求选择其中一个。由指定模块计算出的输出范围后，MpBP 将输出范围与不安全范围（图 4-2 中的红色方块）进行比较，其中不安全区域是由约束生成器由预期性质构造。根据输出范围与不安全范围是否有交集，MpBP 输出 safe 或者 unknown 结果。

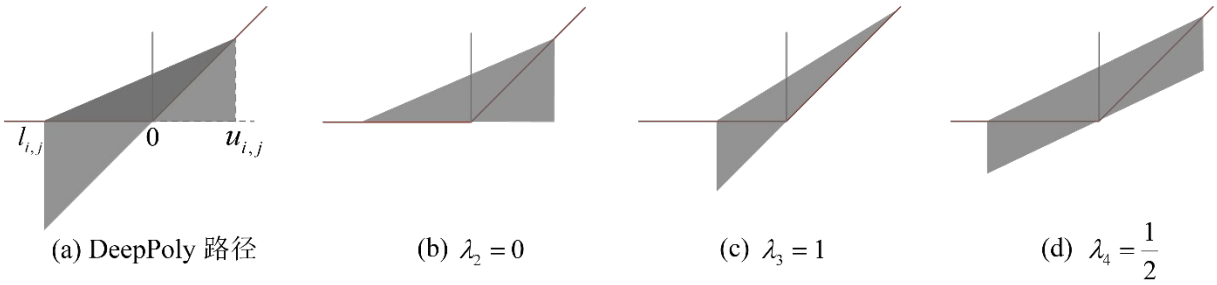
4.2.2 输入输出

MpBP 支持前馈全连接神经网络（feedforward fully-connected neural network，简称 FFNN）和卷积神经网络（convolutional neural network，简称 CNN）的输入，支持最常用的 ReLU 激活函数。输入网络可以是 PyTorch 模型格式 *.pth* 或 *.onnx* 格式。输入中的验证性质表示用户想要验证的性质，它由神经网络的输入 $\mathbf{x}_1$、扰动阈值 θ 和一组用户设定的安全、不安全标签组成。例如，在图 4-2 中，如果想要验证一个 STOP 路标图像即使在扰动阈值 θ 下依旧保持其预测结果，那么使用该 STOP 图像作为 $\mathbf{x}_1$，并将 STOP 标签设置为安全标签（此时其他标签均为不安全）。

对于扰动的类型，MpBP 支持三种类型的 l_p 范数扰动：l_∞、l_1 和 l_2 范数。在路径配置中，用户可以指定用于验证的传播路径的数量。路径数目越多，验证结果越精确，但这种改善不是能够一直持续的。MpBP 使用的默认路径数目为 4，它是由初步实验得到的经验选择。之后可以使用指定的多路径界限传播方法计算给定神经网络关于输入图像 $\mathbf{x}_1$ 和扰动 θ 的输出范围。同时，MpBP 可返回验证为安全或未知，表明输入模型是否满足输入性质。如果返回未知，用户可以尝试更改配置（路径或验证方法），以尝试更精确的验证。

4.2.3 界限传播路径的选择与 GPU 并行化

正如第 3.6.2 节实验所示，随着传播路径数目增加，验证精度的增加并非可持续的。传播路径的增加会引起内存占用的线性增加，因此找到一个尽可能小的路径数目是重要的。MpBP 的默认传播路径数目为 4，它使用对 ReLU 的以下四种上近似方式分别构成四条传播路径：



(a) DeepPoly 路径　(b) $\lambda_2 = 0$　(c) $\lambda_3 = 1$　(d) $\lambda_4 = \frac{1}{2}$

图 4-3 MpBP 默认使用的四条界限传播路径

第一条传播路径利用 DeepPoly[23]的启发式方法，采用最小化局部近似误差的策略，即第 2.3 节的 DeepPoly 路径；当路径数 $M=2$ 时，除第一条路径外，MpBP 利用下界直线的斜率 $\lambda_2=0$ 构成第二条传播路径；对于 $M=3$，第三条传播路径的 $\lambda_3=1$；当 $M\geq 4$ 时，除了上面的前三条路径，MpBP 添加区间 $[0,1]$ 的 $(M-2)$ 分点作为其余 $M-3$ 传播路径的 λ 值。例如，当 $M=4$ 时，第 4 条路径为 $\lambda_4=1/2$。

在初步实验中，若单纯使用一条界限传播路径，经验上效果最为显著的是路径（a），其次是路径（b），然后是路径（d）。综合第 3.6.2 节的实验结果，MpBP 使用以上 4 条默认界限传播路径。

不同路径上的界限传播彼此独立，因此其计算可被并行。理论上而言，通过并行化有限多条界限传播路径上的界限传播，能够在与单条路径相同时间消耗下提高验证精度。MpBP 在机器学习框架 PyTorch 中实现了多路径界限传播。通过将不同路径放入新的张量维度中，它可利用 GPU 的高效张量计算将时间消耗维持在单路径界限传播的水平。在 MpBP 的实现中，给定网络全连接层的界限函数张量形状为：

（batch 大小，路径数目，本层节点数，输入层节点数）

其中 batch 大小为待验证输入的数量。对于前向界限传播，下一层的线性变换矩阵可直接右乘到上述张量，按照 PyTorch 的高维张量乘法得到新的界限函数。高维张量的乘加运算借助 PyTorch 在 GPU 上具有极高的效率，这保障了多路径界限传播方法在实际验证任务中的速度。

4.3 使用案例

作为神经网络鲁棒性验证工具，MpBP 提供基本的命令行用法和扩展的脚本用法，本节将会介绍两种用法，更为详细的用法和参考示例可见 MpBP 网址首页。

4.3.1 命令行接口

用户可在 MpBP 的根目录通过 mpbp.py 接口调用：

```
python mpbp.py  --net <filename>
                --spec <spec file>
                --path <path number>
                --bp <verification method>
                --verbose <output info>
```

图 4-4 MpBP 的命令行接口

其中 <filename> 字段指定待验证神经网络模型文件；<spec file> 文件指定需要被验证的性质，其内容将会在后文介绍；<path number> 指定界限传播路径的数目；<verification method> 指定使用的界限传播方法；<output info> 指定输出的验证结果详细程度，默认值为 False：指定输出统计结果，而不输出每个验证问题的输出范围。

验证性质文件 <spec file> 指定 MpBP 要验证的性质，下面的示例文件指定验证 MNIST 测试集第 1 张图片在 0.01 大小的无穷范数扰动下针对无目标攻击的鲁棒性：

```
dataset = 'MNIST'
x_1 = 0
batch = -1
theta = 0.01
norm = 'inf'
unsafe = 'untarget'
```

图 4-5 MpBP 的验证性质文件示例

使用上述验证性质文件和 4 条路径反向界限传播，可得到如下输出：

```
Running on cuda:0
Bounding method: backward

=============== Reachable region ===============
Image 1
f_0(x_1):  -18.937 <= f_0(x_1+theta) <=   -7.139
f_1(x_1):   -6.983 <= f_1(x_1+theta) <=   -2.559
f_2(x_1):   -7.194 <= f_2(x_1+theta) <=    0.720
f_3(x_1):  -11.366 <= f_3(x_1+theta) <=    0.492
f_4(x_1):  -30.034 <= f_4(x_1+theta) <=  -10.144
f_5(x_1):  -38.756 <= f_5(x_1+theta) <=  -10.372
f_6(x_1):  -36.896 <= f_6(x_1+theta) <=  -14.385
f_7(x_1):    3.691 <= f_7(x_1+theta) <=   12.367
f_8(x_1):  -30.166 <= f_8(x_1+theta) <=  -10.077
f_9(x_1):  -12.881 <= f_9(x_1+theta) <=   -3.071

================== Robustness ==================
Image 1: safe
Time elapsed: 0.08223295211791992
```

图 4-6 MpBP 的输出示例

其中输出表明使用 GPU（Running on cuda: 0）和反向界限传播验证，Reachable region 部分给出输出层节点的取值范围，Robustness 部分给出验证结果和时间统计。

4.3.2 Python 脚本

MpBP 具有良好的封装性，可作为库函数调用。图 4-7 给出了使用 Python 脚本调用 MpBP 库函数计算给定网络输出范围的模板。其用法大致分为四步：加载验证模型、加载验证输入、转化为 MpBP 模型、计算输出范围。其中加载验证模型和输入的用法与 PyTorch 完全相同，这使得已经熟悉 PyTorch 的用户可快速熟悉 MpBP 的灵活使用。用户可通过脚本定义和扩展 MpBP 功能，以适应场景需求。

```python
import torch, torchvision    # 导入 PyTorch 库函数
import torch.nn as nn
from multipath_bp import BoundedModule, BoundedTensor    # 导入 MpBP 库函数
from multipath_bp.perturbations import PerturbationLpNorm

### 第一步：加载待验证模型
def mnist_ffnn():...
model = mnist_ffnn()
checkpoint = torch.load("./mnist_ffnn_10x80.pth", map_location=torch.device('cpu'))
model.load_state_dict(checkpoint)

### 第二步：加载待验证输入
test_data = torchvision.datasets.MNIST("../../examples/vision/data", train=False, download=False,
                                       transform=torchvision.transforms.ToTensor())
N = 100    # Batch 大小
image = test_data.data[:N].reshape(N, 784)
image = image.to(torch.float32) / 255.0    # 转化为 [0,1] 浮点输入
if torch.cuda.is_available():    # 使用 CUDA 并行计算
    image = image.cuda()
    model = model.cuda()

### 第三步：转化为 MpBP 模型
multipath_model = BoundedModule(model, torch.empty_like(image), device=image.device)

### 第四步：定义扰动，计算输出范围
theta, norm = 0.010, float("inf")
ptb = PerturbationLpNorm(norm=norm, theta=theta)
image = BoundedTensor(image, ptb)

for method in ['forward', 'IBP', 'IBP+backward (CROWN-IBP)', 'backward (CROWN)']:    # 选择验证方法
    lb, ub = multipath_model.compute_bounds(x=(image), method=method.split()[0])
    print(lb, ub)    # 打印上下界
```

图 4-7 MpBP 的脚本使用示例

4.4 实验评估

本节将 MpBP 与两个最先进的界限传播工具 LiRPA[5]和 GPUPoly[37]进行实验比较。LiRPA 是第 2 届国际神经网络验证竞赛（VNN-COMP）的获胜者 alpha-beta-CROWN[38]中的界限传播组件；GPUPoly[37]是 ERAN[39]中界限传播方法的 GPU 实现，它是一个长期更新的神经网络验证工具。实验评估在以下三个数据集上进行：

1. 由 MNIST 数据集训练的 FFNN，包含 800 个中间节点；
2. 由 CIFAR-10 数据集训练的 CNN，包含约 2.8×10^5 个中间节点；
3. 由 Tiny ImageNet 数据集训练的 CNN，包含约 4.7×10^5 个中间节点。

所有实验均在 Linux 服务器上运行，服务器配置为两个 Intel Xeon Gold 6132 CPU；一个 NVIDIA Tesla V100 GPU；以及 256 GB 内存。MpBP 使用默认路径数目 4，扰动类型均为常用的 l_∞ 范数。

4.4.1 MpBP 的验证精度

本节比较 MpBP、LiRPA 和 GPUPoly 在三种数据集上的验证精度，实验任务是验证每个数据集中前 100 个正确分类的图像关于不同扰动大小 θ 的鲁棒性。实验结果如表 4-8 所示，表中数据为对应工具成功验证的鲁棒性问题的数量，数字越大，对应工具的验证精度越高，每组实验最大的数字用黑体标出。

表 4-8 MpBP 与 LiRPA、GPUPoly 在三种网络上的精度比较

工具		模型和扰动大小 θ				
		MNIST FFNN				
		0.0014	0.0018	0.0022	0.0026	0.003
FBP	MpBP	**73**	**62**	**51**	**40**	**30**
	LiRPA	69	59	48	33	26
FBBP	MpBP	**86**	**78**	**69**	**58**	**47**
	LiRPA	83	77	66	56	46
		CIFAR-10 CNN			Tiny ImageNet CNN	
		0.001	0.0014		0.001	0.0014
BBP	MpBP	**61**	**38**		**27**	**22**
	LiRPA	56	36		25	19
	GPUPoly	56	36		-	-

容易观察到，对于三种界限传播方法，MpBP 比其他两个工具针对每个扰动大小都更精确。具体地，在第一个网络 MNIST FFNN 上，表 4-8 中将 MpBP 的前向界限传播（FBP）和前向+反向界限传播（FBBP）方法与 LiRPA 中的 FBP 和 FBBP 方法进行比较。GPUPoly 不支持 FBP 或 FBBP，因此不在比较之列。

对于 FBP，给定表 4-8 中的每个扰动阈值，MpBP 均能够比 LiRPA 验证更多的鲁棒性问题。在扰动阈值为 $\theta = 0.0026$ 时，MpBP 成功验证的问题比 LiRPA 多 7（= 40−33）个（总计 100 个问题）。使用更精确的 FBBP 方法，两者都能够验证更多的问题。同样，借助于 FBBP 的多条界限传播路径，MpBP 依旧能够得到优于 LiRPA 的验证结果。

对于最精确的反向界限传播（BBP）验证方法，表 4-8 将 MpBP 在另外两个大型 CNN 上与 LiRPA 和 GPUPoly 进行了比较。在 CIFAR-10 CNN 上，GPUPoly 与 LiRPA 得到相同的验证结果，而 MpBP 对两个扰动阈值分别多验证了 5 和 2 个问题。对于最大的网络 Tiny ImageNet CNN，MpBP 和 LiRPA 能够验证的问题均有所减少。尽管如此，MpBP 比 LiRPA 更具优势，分别多验证 2 个和 3 个问题。表中的“-”表示 GPUPoly 目前不支持 Tiny ImageNet 数据集。综上所述，采用多条界限传播路径的 MpBP 比传统的 LiRPA 和 GPUPoly 具有更高的验证精度。

4.4.2 MpBP 的时间消耗

为衡量 MpBP 的时间消耗，本节选择了上节实验中的 MNIST FFNN 网络和扰动阈值 $\theta = 0.0026$，分别使用以上三个工具验证与上节实验相同的 100 个问题。实验比较三种工具的所有界限传播方法所花费的验证时间。图 4-9 说明了使用不同工具的验证方法验证 100 个鲁棒性问题的总计时间消耗。x 轴列出了工具和方法名称，不同的界限传播类型用不同的颜色分组表示；y 轴表示验证方法花费的验证时间（以秒为单位）。

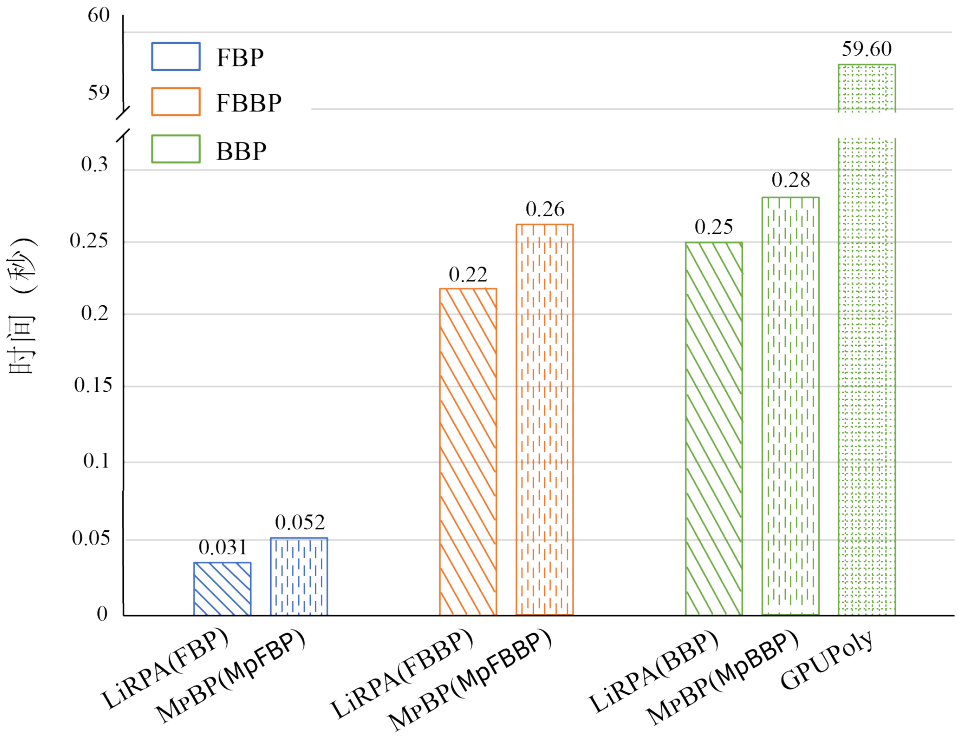


图 4-9 MpBP 与 LiRPA、GPUPoly 的时间消耗比较

结果表明，使用多条界限传播路径的 MpBP 其每种界限传播方法均略慢于相应的 LiRPA 方法，但差异可以忽略不计。对于 BBP，MpBP 能够比同样使用 GPU 加速的 GPUPoly 快约 212 倍。另一方面，从图中也可看出，多路径 FBBP 确实弥补了多路径 FBP 和多路径 BBP 之间的速度差距。

MpBP 在 CNN 网络上也有类似的表现，当选择上节实验的 CIFAR-10 CNN 网络以及扰动大小 $\theta = 0.0014$ 时，MpFBP 平均耗时 0.917 秒，而 FBP 平均耗时 0.717 秒，差异很小。可见利用 GPU 并行也可有效降低多路径界限传播方法的实际时间复杂度，使其达到与单路径方法相当的水平。

综上所述，使用多条界限传播路径的 MpBP 在时间消耗上比 GPUPoly 低很多，而与目前最快速的传统单路径方法 LiRPA 相当。

4.4.3 与 alpha-CROWN 的精度和时间对比

本节将 MpBP 与借助优化方法的 alpha-CROWN[40]，以及传统界限传播方法 GPUPoly 在验证精度和时间消耗两方面进行对比，以定量衡量优化方法对于多路径界限传播方法的精度提升和额外时间代价。Alpha-CROWN 是基于 LiRPA 中反向界

限传播的方法，它将 ReLU 近似的下界斜率 λ 参数化，而非使用具体值。这样鲁棒性验证问题被转化为关于一组 λ 的非凸全局优化问题，这可通过投影梯度下降等方法高效求解。

本实验取 alpha-CROWN 梯度下降的迭代次数为 30 次，MpBP 使用默认配置。实验网络与第 4.4.2 节相同，然后在 MNIST 测试集中选择 50 个输入图像，并使用三种验证方法计算每个图像的鲁棒半径。由第 3.6 节可知，得到的鲁棒半径越大，说明对应方法精度越高。实验结果如图 4-10 散点图所示。

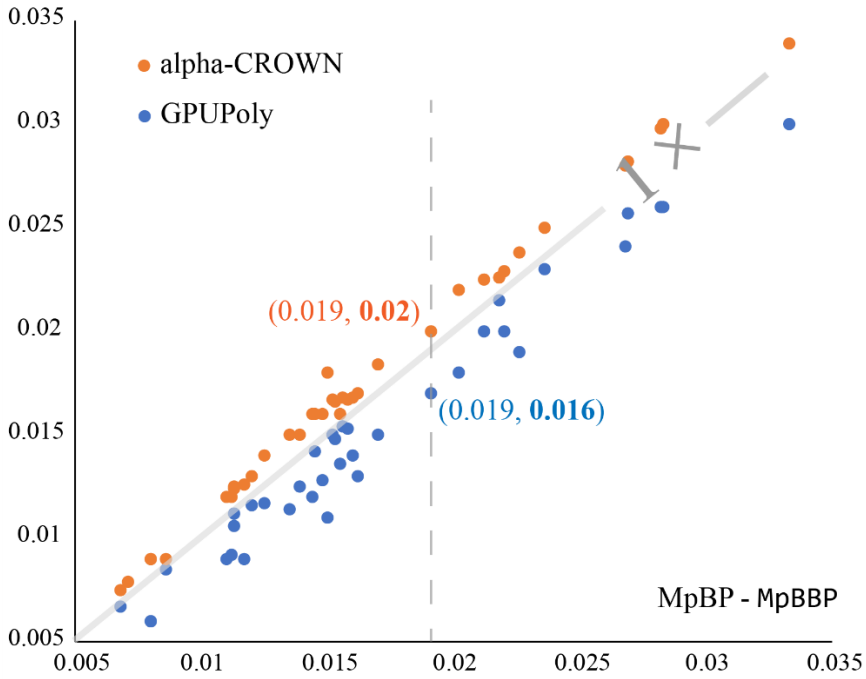


图 4-10 MpBP 与 alpha-CROWN、GPUPoly 得到的鲁棒半径比较

例如，图中垂直虚线上的蓝色圆点和橙色圆点分别表示：当 MpBP 得到一个输入的鲁棒半径为 0.019 时，GPUPoly 得到其鲁棒半径为 0.016，alpha-CROWN 得到其鲁棒半径为 0.02。从图中可知，在此数据集上，借助优化方法的 alpha-CROWN 能够得到的鲁棒半径比多路径界限传播方法 MpBP 高约 0.01~0.02；而 MpBP 比传统的界限传播方法 GPUPoly 高约 0.02。

优化问题的引入会大幅度增加界限传播方法的时间消耗。在本实验中，alpha-CROWN 平均需要 179 秒得到一个鲁棒半径；而完全基于界限传播框架的 MpBP 仅需要 2.17 秒。可见 MpBP 具有极高的精度时间比，且缩小了界限传播方法与优化方法的验证精度差距。

4.5 本章小结

本章将第 3 章反向界限传播路径的概念扩展到其他典型的界限传播方法，除反向界限传播外，还包括前向、前向+反向界限传播。为了提高实用性和验证速度，本章将扩展的多路径界限传播方法实现到机器学习框架 PyTorch 上，并开源为 MpBP 工具。MpBP 在 GPU 上并行化多条界限传播路径，在实验中它能够处理数十万节点的 Tiny ImageNet 卷积网络。同时 MpBP 支持类 PyTorch 脚本用法，具有较高的易用性，一定程度上完备了“训练—验证”流程。

在实验部分，本章将 MpBP 与基于 GPU 的其他界限传播验证工具比较。实验结果说明了 MpBP 能够在与传统界限传播方法相当的时间内显著提高验证精度。本章也将 MpBP 与借助优化方法的 alpha-CROWN 对比，实验结果定量说明了多路径界限传播方法相对于优化方法的精度和时间定位。

第 5 章 界限传播方法的相关技术

本章给出关于界限传播方法更详尽的相关工作和技术讨论。

本文关注如何提高界限传播验证方法的验证精度。关于此问题，过去的一个研究方向是为激活函数设计更精确的线性近似。最简单的界限传播方法是区间界限传播（IBP）[11]，它可看作对 ReLU 等激活函数使用常数作为上下界近似函数；AI^2 工具[41]使用对称多胞体（zonotope）近似激活函数和每层的可达集；DeepZ[42]改进了 zonotope 近似在 ReLU 网络上的适用性，在其实验中它等价于本文中的平行四边形近似，也等价于同期的 Neurify[29]，Fast-Lin[43]以及 Wong 和 Kolter[44]等人采用的近似方式；之后 CROWN 和 DeepPoly 均引入了适应性地选择近似方式来最小化局部近似误差。

关于界限传播方法的类型，最先受到关注的是前向界限传播。例如最早的 IBP 以及之后的 DeepZ 和 Neurify 等，Wong 和 Kolter 从对偶问题的角度给出了反向界限传播方法。随后反向界限传播方法一直作为界限传播方法的主流。

提高界限传播方法验证精度的另外一个研究方向是在界限传播框架内加入代价更高但更精确的方法。最具代表性的是 alpha-CROWN，它将 ReLU 近似的下界斜率参数化，将求解上下界问题转化为非凸全局优化问题，进而可通过投影梯度下降等方法高效求解。从本文的界限传播路径的角度，alpha-CROWN 通过优化方法为界限传播找到一条“较好”的路径。RefinePoly[32]在 DeepPoly 的基础上选择性地混合计算复杂度更高但更精确的 LP 甚至 MILP 以精化结果。类似有 DeepSRGR[31]在 DeepPoly 基础上根据当前输出指导构造 LP 问题以精化验证结果。加入代价更高的精化方法使得问题的求解复杂度取决于计算复杂度较高的方法。本文则完全基于界限传播方法框架，亦可结合 LP 等计算复杂度更高的方法，以精化验证结果。

目前界限传播方法也广泛被用作完备验证的辅助方法。在 beta-CROWN[38]中，作者将分支定界法的 ReLU 分支配置编码到界限传播框架中，这使得基于 BaB 的完备验证能够以界限传播的形式在 GPU 上高效并行化。GCP-CROWN[45]将 MILP 求解器生成的切平面编码到界限传播方法中，同样在界限传播方法的框架内提高了验证精度。从生成切平面的结构而言，此方法可解释为与多节点近似方法[46]一样考虑了同层和不同层节点的取值依赖。

单个神经网络节点的线性近似方法在 LP-ALL[22]中被统一为框架。作者从理论和实验上说明单个节点的线性凸松弛方法与未松弛的原网络验证问题之间有难以弥补的间隙，这一间隙为凸近似间隙（convex barrier）。在其对偶问题的视角下，界限传播方法属于贪心算法的范畴。在界限传播路径的概念下，目前此类方法均使用单条界限传播路径。值得补充的是，LP 近似虽然是单个 ReLU 函数的最精确线性近似，但并不是对 ReLU 神经网络的最精确线性近似，它没有考虑多个节点取值的依赖关系。利用这些依赖信息通常能够得到更精确的近似[47]。但另一方面，通常越精确的近似其时间代价也相对越高。Fast-Lin[43]给出，对于计算给定输入范围下神经网络的可达集这一问题，除非 P=NP，否则不存在近似率为$(1-o(1))\ln N$的多项式时间算法，其中N为给定神经网络的节点数量。

第 6 章 总结

界限传播方法在神经网络验证方法中具有极高的代表性，因为它在精度与时间之间具有较好的折衷，且广泛被作为更精确验证方法的辅助。本文提出界限传播路径的概念，将目前的界限传播方法从单条界限传播路径扩展到多条界限传播路径，形式化地分析了多路径界限传播的可靠性；并将多路径界限传播在 PyTorch 框架上并行化，开发了高效而易用的鲁棒性验证工具。实验结果表明多路径界限传播方法能够有效地提高验证精度，而与传统界限传播方法的时间消耗相当。

本文尚存在不足之处。文中仅讨论 ReLU 激活函数的多路径界限传播方法，实际上本文方法不仅适用于 ReLU 函数，同样适用于其它能够被线性近似的激活函数，如 tanh 和 sigmoid 等，这可作为一个扩展方向。另外，尽管本文方法可以取任意多条界限传播路径，但在每条路径使用相同的上近似方式。如果能够在每条路径上对每个 ReLU 节点有效地选择不同近似方式，理论上验证精度能够获得进一步提升。

参 考 文 献

[1] 董胤蓬, 苏航, 朱军. 面向对抗样本的深度神经网络可解释性分析[J]. 自动化学报, 2022: 75–86.
[2] 王赞, 吴卓, 陈翔. 深度神经网络测试研究综述[J]. 软件学报, 2020: 1255–1275.
[3] Liu C, Arnon T, Lazarus C, *et al*. Algorithms for Verifying Deep Neural Networks[J]. Found. Trends Optim., 2021, 4(3-4): 244-404.
[4] Huang X, Kroening D, Ruan W, *et al*. A Survey of Safety and Trustworthiness of Deep Neural Networks: Verification, Testing, Adversarial Attack and Defence, and Interpretability[J]. Comput. Sci. Rev., 2020: 100-207.
[5] Xu K, Shi Z, Zhang H, *et al*. Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond[C]. Advances in Neural Information Processing Systems. Curran Associates, Inc., 2020: 1129–1141.
[6] Bunel R, Lu J, Turkaslan I, *et al*. Branch and Bound for Piecewise Linear Neural Network Verification[J]. Journal of Machine Learning Research, 2020, 21(42): 1–39.
[7] Huang S, Papernot N, Goodfellow I, *et al*. Adversarial Attacks on Neural Network Policies[C]. 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017.
[8] Sun K, Zhu Z, Lin Z. Enhancing the Robustness of Deep Neural Networks by Boundary Conditional GAN[Z]. arXiv, 2019(2019–02–28).
[9] Balunovic M, Vechev M T. Adversarial Training and Provable Defenses: Bridging the Gap[C]. 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net, 2020.
[10] Raghunathan A, Steinhardt J, Liang P. Certified Defenses against Adversarial Examples[C]. 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018.
[11] Gowal S, Dvijotham K, Stanforth R, *et al*. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models[Z]. arXiv, 2019(2019–08–29).
[12] Li R, Yang P, Huang C-C, et al. Towards practical robustness analysis for DNNs based on PAC-model Learning[C]. Proceedings of the 44th International Conference on Software Engineering. Pittsburgh Pennsylvania: ACM, 2022: 2189–2201.
[13] Fischer M, Baader M, Vechev M T. Scalable Certified Segmentation via Randomized Smoothing[C]. Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event. PMLR, 2021: 3340–3351.
[14] Katz G, Barrett C, Dill D L, *et al*. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks[M]. Computer Aided Verification. Cham: Springer International Publishing, 2017: 97–117.
[15] Katz G, Huang D A, Ibeling D, et al. The Marabou Framework for Verification and Analysis of Deep Neural Networks[M]. DILLIG I, TASIRAN S, eds. Computer Aided Verification. Cham: Springer International Publishing, 2019: 443–452.
[16] Ehlers R. Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks[M]. Automated Technology for Verification and Analysis. Cham: Springer International Publishing, 2017: 269–286.
[17] Dutta S, Jha S, Sanakaranarayanan S, *et al*. Output Range Analysis for Deep Neural Networks[C]. NASA Formal Methods - 10th International Symposium, NFM 2018, Newport News, VA, USA, April 17-19, 2018.
[18] Anderson R, Huchette J, Ma W, et al. Strong Mixed-integer programming formulations for trained neural Networks[J]. Mathematical Programming, 2020, 183(1–2): 3–39.
[19] https://www.gurobi.com/
[20] Lu J, Kumar M P. Neural Network Branching for Neural Network Verification[C]. 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.
[21] Raghunathan A, Steinhardt J, Liang P S. Semidefinite relaxations for certifying robustness to adversarial examples[C]. Advances in Neural Information Processing Systems. Curran Associates, Inc., 2018.
[22] Salman H, Yang G, Zhang H, *et al*. A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks[C]. Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada. 2019: 9832–9842.
[23] Singh G, Gehr T, Püschel M, *et al*. An abstract domain for certifying neural Networks[J]. Proceedings

of the ACM on Programming Languages, 2019, 3(POPL): 1–30.
[24] Zhang H, Weng T-W, Chen P-Y, *et al*. Efficient Neural Network Robustness Certification with General Activation Functions[J]. Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montreal, Canada.
[25] Zhang H, Chen H, Xiao C, *et al*. Towards Stable and Efficient Training of Verifiably Robust Neural Networks[C]. 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.
[26] https://pytorch.org/
[27] Le Y, Yang X. Tiny imagenet visual recognition challenge.
[28] Huang X, Kwiatkowska M, Wang S, et al. Safety Verification of Deep Neural Networks[M]. MAJUMDAR R, KUNČAK V, eds. Computer Aided Verification. Cham: Springer International Publishing, 2017: 3–29.
[29] Wang S, Pei K, Whitehouse J, *et al*. Efficient Formal Safety Analysis of Neural Networks[C]. Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. 2018: 6369–6379.
[30] Wang S, Pei K, Whitehouse J, *et al*. Formal Security Analysis of Neural Networks using Symbolic Intervals[C]. 27th USENIX Security Symposium (USENIX Security 18). Baltimore, MD: USENIX Association, 2018: 1599–1614.
[31] Yang P, Li R, Li J, *et al*. Improving Neural Network Verification through Spurious Region Guided Refinement[J]. Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021.
[32] Singh G, Gehr T, Püschel M, *et al*. Robustness Certification with Refinement[C]. International Conference on Learning Representations. 2019.
[33] https://github.com/formes20
[34] Julian K D, Lopez J, Brush J S, *et al*. Policy compression for aircraft collision avoidance systems[C]. 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC). Sacramento, CA, USA: IEEE, 2016: 1–10.
[35] http://yann.lecun.com/exdb/mnist/
[36] https://www.cs.toronto.edu/~kriz/cifar.html
[37] Müller C, Serre F, Singh G, *et al*. Scaling Polyhedral Neural Network Verification on GPUs[C]. Proceedings of Machine Learning and Systems 2021, MLSys 2021, virtual, April 5-9, 2021.
[38] Wang S, Zhang H, Xu K, *et al*. Beta-CROWN: Efficient Bound Propagation with Per-neuron Split Constraints for Complete and Incomplete Neural Network Verification[J]. arXiv, 2021.
[39] https://github.com/eth-sri/eran
[40] Xu K, Zhang H, Wang S, *et al*. Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers[C]. 9th International Conference on Learning Representations, ICLR, 2021, Virtual Event, Austria, May 3-7, 2021.
[41] Gehr T, Mirman M, Drachsler-Cohen D, *et al*. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation[C]. 2018 IEEE Symposium on Security and Privacy (SP). San Francisco, CA: IEEE, 2018: 3–18.
[42] Singh G, Gehr T, Mirman M, *et al*. Fast and Effective Robustness Certification[C]. Advances in Neural Information Processing Systems. Curran Associates, Inc., 2018.
[43] Weng L, Zhang H, Chen H, *et al*. Towards Fast Computation of Certified Robustness for ReLU Networks[C]. Proceedings of the 35th International Conference on Machine Learning. PMLR, 2018: 5276–5285.
[44] Wong E, Kolter J Z. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope[J]. Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmassan, Stockholm, Sweden, July 10-15, 2018.
[45] Zhang H, Wang S, Xu K, *et al*. General Cutting Planes for Bound-Propagation-Based Neural Network Verification[Z]. arXiv, 2022(2022–08–11).
[46] Müller M N, Makarchuk G, Singh G, *et al*. PRIMA: Precise and General Neural Network Certification via Multi-Neuron Convex Relaxations[J]. Proc. ACM Program. Lang, 2022: 1-33.
[47] Xie X, Chen B, Liu Y, et al. Proteus: Computing disjunctive loop summary via path dependency Analysis[C]. Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Seattle WA USA: ACM, 2016: 61–72.

致　谢

深圳天凉了。

窗未合紧，从窗缝倾泻而入的一道冷风，让我想起过去的很多个冬天。这将是我在深圳大学的第三个冬天。

回忆里的这段时光安静、深沉，少许画面，长篇独白。快乐厚重而柔和，多是孤独与劳累之后艰难的快乐，八分宽慰，两分欢欣。还有很多难以忘怀的帮助，每想起时，感激之外，更为感动。

谢谢嘉祥老师和晓牧老师。我将永远庆幸能遇到你们。一生不过一两位导师，却影响一生，我将永远铭记你们的帮助。我写不出很多感谢的话，也觉得感谢是多余，心中只有一句：何其有幸。

谢谢厚华、玉钊学长，谢谢师弟师妹。谢谢你们的温暖，谢谢你们没有介意我说过的很多废话。我生活单调，你们几乎是我硕士生活的回忆。

谢谢深圳大学。谢谢绸缎般的晨光、悄悄滑进窗的余晖，谢谢木棉、海棠、紫薇、鸢尾，谢谢冬日暖阳、仲夏弯月，谢谢跟忙碌城市说的早安晚安。时间转得太快，尚未起笔时，我仍以为有很多可以浪费的时辰。

过去总在回忆中显露美好，当时只觉是寻常。

谢谢自己。更多时候我在与自己对话，述说放大的情绪，追问模糊的意义。我曾将自己从绝望中拯救，也怂恿自己奋不顾身，很多自以为明白的事情一遍又一遍看不清。谢谢回忆里的努力和坚持，谢谢认真经历悲与喜的自己。

还有很多已经来不及说或是说不出口的谢谢，谢谢你们的并肩或是擦肩。

我愧对父母。直到此刻我还不能真切地明白，对他们而言，我是最重要的人。或许伟大的爱从来是不对等的，太多理所应当让人觉得平常。明知聚少离多，我依旧说不出关心，我比十年前更加小心翼翼。只是有时我在梦里哭，醒来继续悲伤。恩重如山，我说不出写不出谢谢，我只有愧疚，我空有愧疚。

落笔于此，内心慌张，悲欣交集。言辞凌乱，情深意重。我羞于用笔墨渲染情绪，寥寥数句，写了又删，删了又写，终于只留下：

愿你们平安喜乐，万事胜意。

郑烨

二零二二年十二月七日・夜

攻读硕士学位期间的研究成果

[1] **郑烨**, 施晓牧, 刘嘉祥. 基于多路径回溯的神经网络验证方法. **软件学报**, 2022, 33(7): 2464~2481.

[2] (上述软件学报论文推荐翻译出版) **Ye Zheng**, Xiaomu Shi, Jiaxiang Liu. Multi-path Back-propagation Method for Neural Network Verification. International Journal of Software and Informatics **(IJSI)**, 2022,12(4): 377~401.

[3] **Ye Zheng**, Jiaxiang Liu, and Xiaomu Shi. MpBP: Verifying Robustness of Neural Networks with Multi-path Bound Propagation. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering **(ESEC/FSE 2022)**, 1692~1696.