CNN-Cert: 一个验证卷积神经网络鲁棒性的高效框架

Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu and Luca Daniel

摘要

由于深度神经网络的成功及其对对抗性扰动意想不到的脆弱性，验证神经网络的鲁棒性引起了人们的极大兴趣和关注。虽然找到 ReLU 激活神经网络的最小对抗扰动已被证明是一个 NP 完全问题，但我们仍然可以尝试找到一个非平凡的最小扰动下界作为可证明的鲁棒性保证。以前的大多数工作只关注简单的全连接层，并局限于 ReLU 激活。因此，文章提出了一个通用且高效的框架来证明通用卷积神经网络上的鲁棒性。

本框架通用且高效，它可以处理各种架构，包括卷积层、最大池化层、批归一化层、残差块以及通用激活函数; 通过利用卷积层的特殊结构，实现了与目前最先进的认证算法（如 Fast-Lin、CROWN）相比的 17 倍和 11 倍的速度提升，同时算法仍然能够获得类似甚至更好的验证边界。此外，CNN-Cert 推广了最先进的算法，如 Fast-Lin 和 CROWN。

实验结果证明，本方法在边界质量和速度方面都优于最先进的基于下界的认证算法。

关键词：卷积神经网络；鲁棒性验证

1 引言

在神经网络中，评价模型的鲁棒性的方法分为两类：基于博弈的方法和基于验证的方法。前者是通过实证验证指定的攻击，看模型是否能抵御这种攻击，也就是攻击可知的，对于未测试过的攻击它不一定能保证这样的鲁棒性。所以我们主要讨论后面这种基于验证的方法，它给出的鲁棒性认证适用于威胁模型下任何可能攻击。

这类基于验证的方法主要目标是找出最小扰动，也就是当所受到的扰动小于最小扰动时，模型不会被愚弄，依旧能够正确输出（原输入周围的所有扰动输入都具有与原输入相同的分类结果）。但是找出神经网络的最小对抗扰动已经被证明是一个 NP 完全问题，也就是说它很难甚至不能在多项式时间内求解。所以我们转而更保守地去求它的下界，试图找到一个非平凡的最小扰动下界。

2 相关工作

前人在这方面已经有很多相关研究，但目前已有的工作普遍有两个缺陷，一个是不能支持不同的网络架构和激活函数，一个是局限于全连接神经网络，在处理卷积神经网络时只是简单地把卷积层转换回全连接层。其中，Fast-Lin 和 Fast-Lip 为全连接神经网络引入了非平凡的鲁棒性边界，CROWN 把范围扩展到通用激活函数，而本文用于通用卷积神经网络体系结构。

2.1 对抗攻击与防御

在目标模型完全透明的白盒设置中，最近的工作展示了对由神经网络支持的机器学习应用程序的对抗攻击，包括对象识别[1]、图像字幕[2]、机器翻译[3]和图学习[4]。甚至在黑盒设置下，对手只被允许访问模型输出，不允许访问模型内部，而对抗攻击仍然是可信的[5][6][7][8]。为了提高神经网络的鲁棒性，带有对抗攻击的对抗训练是迄今为止最有效的策略之一，显示出强大的实证防御性能[8][9]。此外，基于验证的方法已经验证了具有对抗训练的神经网络确实可以提高鲁棒性[10][11]。

2.2 神经网络的鲁棒性验证

在范数球有界的威胁模型下，对于具有 ReLU 激活函数的神经网络，尽管最小对抗扰动给出了最佳的认证鲁棒性，但由于其 NP 完全的复杂性难以求解[12]。作为替代，更可能去求解最小扰动的非平凡下界作为鲁棒性证明，但代价是得到的鲁棒性证明更加保守。一些的分析下界仅取决于模型权重[1][13][14][15]，但它们通常太松散，无法发挥作用，或局限于 1 或 2 个隐藏层。神经网络的鲁棒性可以被在 ReLU 激活[10][16]和通用激活[17]上高效地认证，但主要是在具有全连接层的模型上。也可以应用于不同的激活函数，但作为计算效率之间的权衡，它们的边界质量可能下降很多[18]。

3 本文方法

3.1 本文方法概述

文章提出的 CNN-Cert 是一个神经网络的鲁棒性认证框架，它可以处理各种各样的构建块，包括卷积层、池化层、批归一化和残差块，也支持各种通用激活函数，边界以卷积形式表示，支持任何可以由卷积函数限制的构建块。算法的中心思想是在神经网络中的非线性操作上应用线性边界技术，使用输入的两个线性函数上下限整个网络，如果输入被限定在一定范围内，那么输出也可以被限定在一个范围内。

文章考虑了四个常用构建块，分别是带有激活函数的卷积层、残差块、批归一化块和池化块。推导显式网络输出边界的关键步骤是考虑每个构建块的输入/输出关系，用红色箭头标记。激活层可以是通用激活，但这里用 ReLU 表示。如图 1所示。

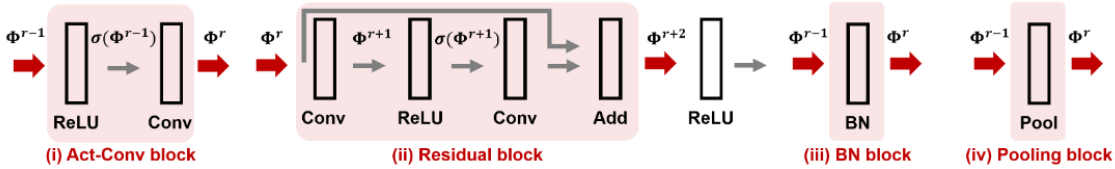


图 1: 常用构建块

3.2 处理常用构建块

3.2.1 处理非线性激活函数和卷积层

其中 ϕ^{r-1} 是激活层的输入，ϕ^r 是卷积层的输出。输入和输出的关系是 $\phi^r = W^r * \sigma(\phi^{r-1}) + b^r$，也就是输入经过激活函数，和权重进行卷积操作，再加上偏置。给激活函数添加两个线性边界，即 $\alpha_L(y+\beta_L) \leqslant \alpha(y) \leqslant \alpha_U(y+\beta_U)$，当输入 y 被限制在某个范围内时，可以找到恰当的 α 和 β 满足这个式子。当限制了第一个公式中的激活函数时，就可以得到这个构建块的边界，即 $A^r_{L,act} * \phi^{r-1} + B^r_{L,act} \leqslant \phi^r \leqslant A^r_{U,act} * \phi^{r-1} + B^r_{U,act}$。

另外，这个操作可以迭代地进行，把这个操作用到第 $r-1$ 层，用第 $r-2$ 层的输出上下限第 $r-1$ 层的输出，一直到第 0 层，也就是最后可以用输出数据表示出第 r 层的上下界, 即 $A^0_{L,conv} * x + B^0_L \leqslant \phi^r(x) \leqslant A^0_{U,conv} * x + B^0_U$。

3.2.2 处理残块操作

对于残差块，设 ϕ^{r+2} 为残差块的输出，ϕ^{r+1} 是第一个卷积层的输出，ϕ^r 是残差块的输入，可以得到 $\phi^{r+1} = W^{r+1} * \phi^r + b^{r+1}$，$\phi^{r+2} = W^{r+2} * \sigma(\phi^{r+1}) + b^{r+2} + \phi^r$。这些和前面对激活函数和卷积层的处理很类似，只是这里多了一个 ϕ^r 是残差块里快接连接的体现。最终得到与卷积层形式类似的边界

$A_{L,res}^{r+2} * \phi^r + B_{L,res}^{r+2} \leqslant \phi^{r+2} \leqslant A_{U,res}^{r+2} * \phi^r + B_{U,res}^{r+2}$。

3.2.3 处理批归一化

对于批归一化，ϕ^{r-1} 是输入，ϕ^r 是输出。它先减均值除方差，再进行尺度变换和偏移，调整数据分布变化，也就是 $\phi^r = \gamma_{bn}\frac{\phi^{r-1} - \mu_{bn}}{\sqrt{\sigma_{bn}^2 + \varepsilon_{bn}}} + \beta_{bn}$。设置 $A_{U,bn}^r = A_{L,bn}^r = \frac{\gamma_{bn}}{\sqrt{\sigma_{bn}^2 + \sigma_{bn}}}$，$B_{U,bn}^r = B_{L,bn}^r = -\gamma_{bn}\frac{\mu_{bn}}{\sqrt{\sigma_{bn}^2 + \varepsilon_{bn}}} + \beta_{bn}$，整理公式，依旧可以得到类似形式的输入输出关系 $A_{L,bn}^r * {}^{r-1} + B_{L,bn}^r \leqslant \phi^r \leqslant A_{U,bn}^r * \phi^{r-1} + B_{U,bn}^r$。

3.2.4 处理池化操作

这里选择的是最大池化，即 $\phi_n^r = \max_{S_n} \phi_{S_n}^{r-1}$，$S_n$ 表示与第 n 个输出相关联的池化输入索引集，可得 $A_{L,pool}^r * \phi^{r-1} + B_{L,pool}^r \leqslant \phi^r \leqslant A_{U,pool}^r * \phi^{r-1} + B_{U,pool}^r$。

3.3 计算神经网络输出的全局边界

上面给出了单个构建块的显式输出界限，当各个构建块的输入被限定在一定范围内时，构建块的输出也可以被两个线性函数限定。因为神经网络可以被视为是构建块的串联，当前构建块的输入是上一个构建块的输入，所以可以把边界从最后一个构建块反向传播给第一个构建块，得到网络输入和输出的上下界 $A_L^0 * x + B_L^0 \leqslant \phi^m(x) \leqslant A_U^0 * x + B_U^0$。

又因为扰动以 L_p 范数为边界，即把扰动限制在一个以 x_0 为中心，ε 为半径的范数球内，所以又可以得到全局上下界 $\eta_{j,U} = \varepsilon\|vec(A_U^0)\|_q + A_U^0 * x_0 + B_U^0$，$\eta_{j,L} = \varepsilon\|vec(A_L^0)\|_q + A_L^0 * x_0 + B_L^0$。

3.4 计算认证下界

假设 c 是输入的预测分类，t 是目标分类，可以通过检查 $\phi_c^m(x) - \phi_t^m(x) > 0$ 是否成立来判断模型分类结果。应用上面计算出的全局边界，也就是可以先给出一个最大输入扰动，然后判断 $\eta_{c,L} - \eta_{t,U} > 0$ 是否为真，如果条件为真就加大扰动，为假就降低扰动。

3.5 激活函数的上下界

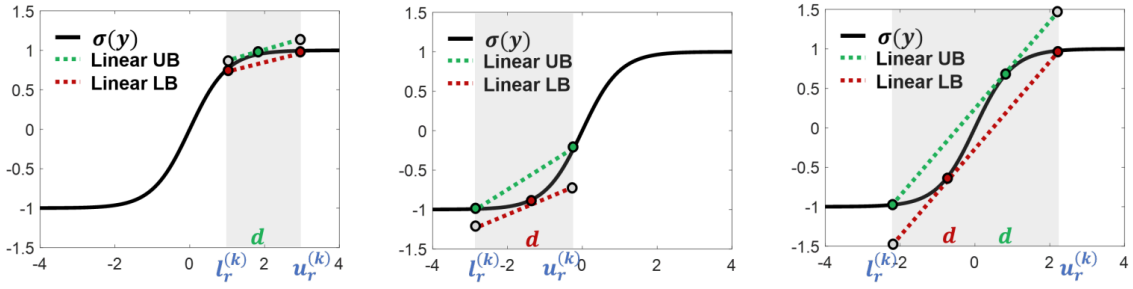
3.5.1 分段激活函数

根据 $l_r^{(k)}$ 和 $u_r^{(k)}$ 的符号，定义第 k 层神经元集 $[n_k]$ 的分割 $\{S_k^+, S_k^\pm, S_k^-\}$，使第 k 层中的每个神经元都只属于三个集合中的一个。$S_k^+, S_k^\pm, S_k^-$ 的正式定义为 $S_k^+ = \{r \in [n_k] \mid 0 \leqslant l_r^{(k)} \leqslant u_r^{(k)}\}$，$S_k^\pm = \{r \in [n_k] \mid l_r^{(k)} < 0 < u_r^{(k)}\}$，$S_k^- = \{r \in [n_k] \mid l_r^{(k)} \leqslant u_r^{(k)} \leqslant 0\}$，这样的分割集体现了 tanh、sigmoid 和 arctan 函数的曲率和 ReLU 的激活状态。对于每个分割集里的神经元，根据 $l_r^{(k)}$ 和 $u_r^{(k)}$ 定义相关的上界 $h_{U,r}^{(k)}$ 和下界 $h_{L,r}^{(k)}$。

3.5.2 tanh/sigmoid/arctan 的上下界

三类函数都是一边凸（$y < 0$），另一边凹（$y > 0$），因此，可以用同样的规则求出 $h_{U,r}^{(k)}$ 和 $h_{L,r}^{(k)}$。我们称 $(l_r^k, \sigma(l_r^{(k)}))$ 为左端点，称 $(u_r^{(k)}, \sigma(u_r^{(k)}))$ 为右端点。对于 $r \in S_k^+$，因为 $\sigma(y)$ 是凹的，我们可以令 $h_{U,r}^{(k)}$ 为 $\sigma(y)$ 在点 $d \in [l_r^{(k)}, u_r^{(k)}]$ 的任意切线，令 $h_{L,r}^{(k)}$ 为 $\sigma(y)$ 经过两个端点。对于 $r \in S_k^-$，与 $r \in S_k^+$ 同理。对于 $r \in S_k^\pm$，令 $h_{U,r}^{(k)}$ 为 $\sigma(y)$ 为经过左端点和 $(d, \sigma(d))$ 的切线，其中 $d \geqslant 0$，$h_{L,r}^{(k)}$ 为 $\sigma(y)$ 为经过右端点和 $(d, \sigma(d))$ 的切线，其中 $d \leqslant 0$。超越函数中的 d 值可以用二分法找到。

tanh 的上下界如图 2所示，绿线是上界 $h_{U,r}^{(k)}$，红线是下界 $h_{L,r}^{(k)}$。



(a) $r \in \mathcal{S}_k^+$ (b) $r \in \mathcal{S}_k^-$ (c) $r \in \mathcal{S}_k^\pm$

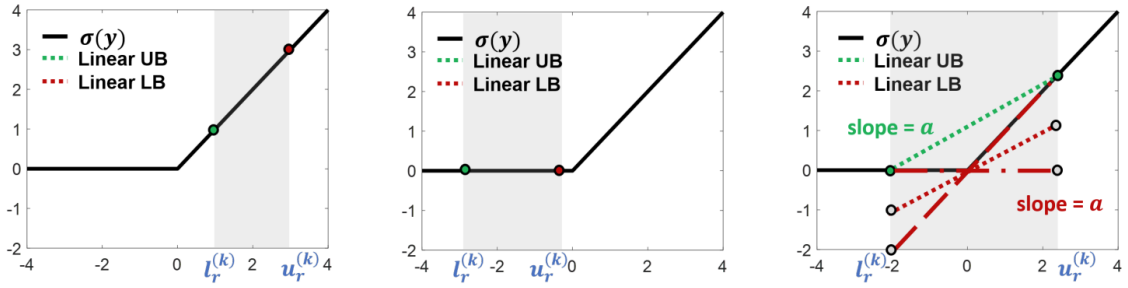
图 2: $\sigma(y) = tanh$ 的线性上下界

3.5.3 ReLU 的上下界

对于 ReLU 激活，$\sigma(y) = max(0, y)$。如果 $r \in S_k^+$，那么 $\sigma(y) = y$，可得 $h_{U,r}^{(k)} = h_{L,r}^{(k)} = y$；如果 $r \in S_k^-$，那么 $\sigma(y) = 0$，可得 $h_{U,r}^{(k)} = h_{L,r}^{(k)} = 0$；如果 $r \in S_k^\pm$，可得 $h_{U,r}^{(k)} = \frac{u_r^{(k)}}{u_r^{(k)} - l_r^{(k)}}(y - l_r^{(k)})$，$h_{L,r}^{(k)} = ay, a \leqslant a \leqslant 1$。这里有两种选择 a 的策略，可以令 $a = \frac{u_r^{(k)}}{u_r^{(k)} - l_r^{(k)}}$，也可以自适应选择 a：当 $u_r^{(k)} \geqslant |l_r^{(k)}|$ 时，令 $a = 1$；当 $u_r^{(k)} \leqslant |l_r^{(k)}|$ 时，令 $a = 0$。这样，下界 $h_{L,r}^{(k)} = ay$ 和 $\sigma(y)$ 之间的面积（即下界和 ReLU 函数之间的间隙）总是最小的。

$\sigma(y) = ReLU$ 的线性上下界如图 3所示，对于 (a) 和 (b)，线性上界和下界正是灰色阴影区域中的函数 $\sigma(y)$。对于（c），众多下界选择中的三个例子分别是：点划线表示的 $h_{L,r}^{(k)} = 0$、虚线表示的 $h_{L,r}^{(k)} = y$、点线表示的 $h_{L,r}^{(k)} = \frac{u_r^{(k)}}{u_r^{(k)} - l_r^{(k)}} y$。

ReLU 的上下界如图 3所示，绿线是上界 $h_{U,r}^{(k)}$，红线是下界 $h_{L,r}^{(k)}$。



(a) $r \in \mathcal{S}_k^+$ (b) $r \in \mathcal{S}_k^-$ (c) $r \in \mathcal{S}_k^\pm$

图 3: $\sigma(y) = ReLU$ 的线性上下界

4 复现细节

4.1 与已有开源代码对比

复现过程中参考了文章作者公开发布的源码，在复现其源码的基础上对其计算激活函数的上下界的方法进行了改进，使用了更加紧密的方法上下限激活函数，由于各构建块的显式线性边界会逐层传播，对非线性激活函数的上下限方式极大地影响了整个网络上下界精度，通过证明神经网络在更大的扰动下界内具有鲁棒性来加强卷积神经网络的鲁棒性验证，最大达到了 15% 的提升。此外，还借用了 Fast-Lin、CLEVER 等框架源码进行对比试验。

Algorithm 1 General and efficient verification framework for certifying robustness of CNNs.

Input: weights and biases of m layers: $W^{(1)}, \cdots, W^{(m)}, b^{(1)}, \cdots, b^{(m)}$ original class c,target class j
Output: certified lower bound $\tilde{\varepsilon_j}$
Function `CNNCert(`x_0, p, ε_o`)`**:**
  while ε *has not achieved a desired accuracy and iteration limit has not reached* **do**
    for $k \leftarrow 1$ **to** m **do**
      $l^{(0)}, u^{(0)} \leftarrow$ COMPUTETWOSIDEBOUNDS($x_0, \varepsilon, p, l^{(1:k-1)}, u^{(1:k-1)}, k$)
    end
    if $l^{(m)} > 0$ **then**
      ε is a lower bound; increase ε using a binary search procedure
    else
      ε is not a lower bound; decrease ε using a binary search procedure
    end
  end
  $\tilde{\varepsilon_j} \leftarrow \varepsilon$
End Function
Function `COMPUTETWOSIDEBOUNDS(`x_0, ε_o, p, $l^{1:m'-1}$, $u^{1:m'-1}$, m'`)`**:**
  if $m' = 1$ **then**
    $A^{(0)} \leftarrow W^{(1)}$
  else
    for $k \leftarrow m' - 1$ **to** 1 **do**
      if $m' = 1$ **then**
        Construct diagonal matrix $D^{(k)} \in \mathbb{R}^{n_k \times n_k}$ using $l^{(k)}, u^{(k)}$.
        $A^{(m'-1)} \leftarrow W^{(m')} D^{(m'-1)}$
      else
        $A^{(k)} \leftarrow A^{(m'-1)} A^{(k)}$
      end
      $T^{(k)} \leftarrow 0, H^{(k)} \leftarrow 0$
      for *all* $r \in \mathcal{I}_k$ **do**
        for $j \leftarrow 1$ **to** n_k **do**
          if $A^{(k)}_{j,r} > 0$ **then**
            $T^{(k)}_{r,j} \leftarrow l^{(k)}_r$
          else
            $H^{(k)}_{r,j} \leftarrow l^{(k)}_r$
          end
        end
      end
    end
  end
  for *j=1* **to** $n_{m'}$ **do**
    $v_j \leftarrow A^{(0)}_{j,:} x_0 + b^{(m')}_j, \mu^+_j \leftarrow 0, \mu^-_j \leftarrow 0$
    for $k = 1$ **to** $m' - 1$ **do**
      $\mu^+_j \leftarrow \mu^+_j - A^{(k)}_{j,:} T^{(k)}_{:,j}, \mu^-_j \leftarrow \mu^-_j - A^{(k)}_{j,:} H^{(k)}_{:,j}$
      $v_j \leftarrow v_j + A^{(k)}_{j,:} b^{(k)}$
    end
    $\gamma^U_j \leftarrow \mu^+_j + v_j + \varepsilon \|A^{(0)}_{j,:}\|_q$
    $\gamma^L_j \leftarrow \mu^-_j + v_j + \varepsilon \|A^{(0)}_{j,:}\|_q$
  end
  return γ^L, γ^U
End Function

4.2 实验环境搭建

实验在 python3.6 和 TensorFlow v1.12 环境下进行。环境中需要安装以下包：pillow、numpy、scipy、pandas、h5py、tensorflow、numba、posix_ipc、matplotlib。可以运行以下 conda 命令创建虚拟环境，并在创建的虚拟环境中安装所需要的包。

```
conda create --name cnncert python=3.6
conda activate cnncert
conda install pillow numpy scipy pandas h5py tensorflow=1.12 numba posix_ipc
   matplotlib
```

如果需要运行 LP/Dual-LP 方法，需要安装 Gurobi 及其相关的 python 库 gurobipy 及其许可证。

```
conda config --add channels http://conda.anaconda.org/gurobi
conda install gurobi
```

4.3 界面分析与使用说明

运行 converting_model.py，将预训练的 CNN 模型转换为 MLP 模型，以便与 Fast-Lin 进行比较。生成 converting_model.txt 文件，其中保存转换模型的结果信息。

```
$ python converting_model.py
```

运行 setup.py，复现实验。生成 result.txt 文件，其中保存鲁棒性证明和平均运行时间信息。

```
$ python setup.py
```

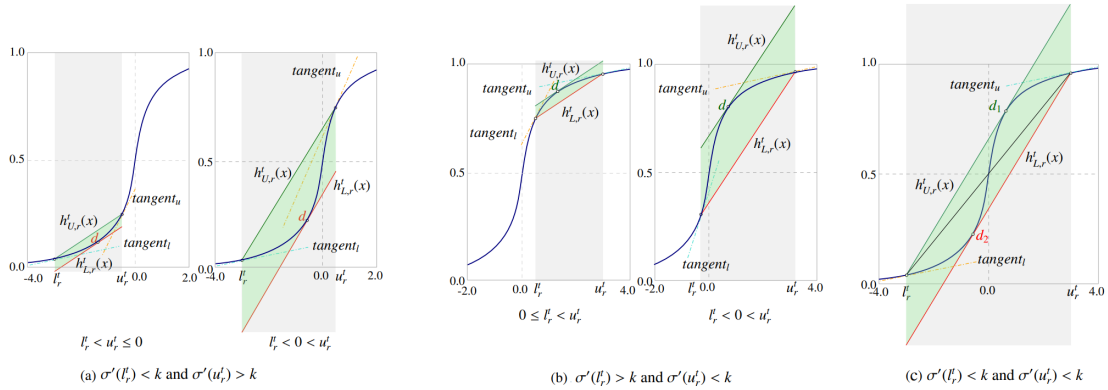
4.4 创新点

从上述分析可见，基于 $l_r^{(k)}$ 和 $u_r^{(k)}$ 分割激活函数对于限界给定的激活函数很有用。有多种方法分割激活函数和定义分割集（如根据 $l_r^{(k)}$、$u_r^{(k)}$ 的值，而不是符号），可以将其合并到框架中，为新的分割集提供相应的显式输出边界。由于各构建块的显式线性边界会逐层传播，对非线性激活函数的上下限方式极大地影响了整个网络上下界精度。所以我对上下限非线性激活函数的方式做出了改进，使用了更加紧密的方法上下限激活函数。

对于 sigmoid-like 函数，文章根据激活函数前的值分出三种情况，这里我做出了更细粒度的分割。具体操作为在函数图像上先连接两个端点，将这条连线的斜率称为 k，即 $k=\dfrac{\sigma(u_r^t)-\sigma(l_r^t)}{u_r^t-l_r^t}$。然后比较两个端点处函数的斜率大小（$\sigma'(l_r^t)$、$\sigma'(u_r^t)$）和 k 值的大小，分出三种情况：（a）$\sigma'(l_r^t)<k,\sigma'(u_r^t)>k$；（b）$\sigma'(l_r^t)>k,\sigma'(u_r^t)<k$；（c）$\sigma'(l_r^t)<k,\sigma'(u_r^t)<k$。

对于第一种情况，与原文一样先连接端点作为上界，即 $h_{U,r}^t=k(x-l_r^t)+\sigma(l_r^t)$，但是对于下界，文章中简单地选取了端点中点作为切点，而我改进为了选择一条斜率与上界一致的切线，即 $h_{L,r}^t=k(x-d)+\sigma(d)$（$\sigma'(d)=k,l_r^t<d<u_r^t$），这是一个不同。第二种情况与第一种情况类似，即 $h_{U,r}^t=k(x-d)+\sigma(d)$（$\sigma'(d)=k,l_r^t<d<u_r^t$），$h_{L,r}^t=k(x-l_r^t)+\sigma(l_r^t)$。第三种情况的处理方式与原文一致，不做改变，即 $h_{U,r}^t=\sigma'(d_1)(x-l_r^t)+\sigma(l_r^t)$（$\sigma'(d_1)=\dfrac{\sigma(d_1)-\sigma(l_r^t)}{d_1-l_r^t}$），$h_{L,r}^t=\sigma'(d_2)(x-u_r^t)+\sigma(u_r^t)$（$\sigma'(d_2)=\dfrac{\sigma(d_2)-\sigma(u_r^t)}{d_2-u_r^t}$）。

可以看到，图 4中的（b）$l_r^t<0<u_r^t$ 情况在文章中会被归为第三类，但是在改进后的方法中被归进第一类，以更紧密的方式上下限激活函数。



(a) $\sigma'(l_r^t) < k$ and $\sigma'(u_r^t) > k$ (b) $\sigma'(l_r^t) > k$ and $\sigma'(u_r^t) < k$ (c) $\sigma'(l_r^t) < k$ and $\sigma'(u_r^t) < k$

图 4: 改进后 $\sigma(y) = sigmoid$ 的线性上下界

5 实验结果分析

在具有五个过滤器、三个卷积核的四层卷积神经网络以及具有两个残差块的残差网络两类网络上进行了计算网络认证下界的实验。实验分别考虑了 ReLU、sigmoid、tanh 和 arctan 四种不同的激活函数，以及 L_1、L_2、L_∞ 三种不同的范数。

图 10显示了包括改进前后的认证下界和平均运行时间在内的实验结果。可以看到，改进后的方法总是返回比原方法更大的下界，最大达到了 15% 的提升。改进后的方法在残差网络上有更好的性能，因为后层会反复使用前层的输出。同时，改进后的方法在 L_∞ 范数上效果更好。在时间效率方面，在相同的网络架构下，改进前后的平均耗时几乎相同。

实验证明，本方法确实可以高效地为所有被测试的激活函数找到非平凡下界，并且计算通用激活函数的认证下界不会导致显著的计算损失。使用更紧密的方法上下限激活函数能够有效提升整个网络的上下界精度。与卷积神经网络相比，残差网络的计算下界对激活函数近似的精度更敏感。

```
CNN-Cert
model = models/mnist_cnn_4layer_5_3, norm = i, activation = ReLU
robustness = 0.04920, runtime = 0.92s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = i, activation = sigmoid
robustness = 0.06536, runtime = 0.99s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = i, activation = tanh
robustness = 0.02226, runtime = 1.18s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = i, activation = arctan
robustness = 0.02178, runtime = 1.00s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = 2, activation = ReLU
robustness = 0.17936, runtime = 0.93s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = 2, activation = sigmoid
robustness = 0.34444, runtime = 1.00s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = 2, activation = tanh
robustness = 0.15177, runtime = 1.32s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = 2, activation = arctan
robustness = 0.13913, runtime = 1.05s
```

图 5: 原方法计算出的 CNN 认证下界

```
CNN-Cert
model = models/mnist_resnet_2, norm = i, activation = ReLU
robustness = 0.03961, runtime = 1.58s
-----------------------------------------------------------------
model = models/mnist_resnet_2, norm = i, activation = sigmoid
robustness = 0.06083, runtime = 1.65s
-----------------------------------------------------------------
model = models/mnist_resnet_2, norm = i, activation = tanh
robustness = 0.06046, runtime = 1.67s
-----------------------------------------------------------------
model = models/mnist_resnet_2, norm = i, activation = arctan
robustness = 0.03534, runtime = 1.64s
-----------------------------------------------------------------
model = models/mnist_resnet_2, norm = 2, activation = ReLU
robustness = 0.15282, runtime = 1.65s
-----------------------------------------------------------------
model = models/mnist_resnet_2, norm = 2, activation = sigmoid
robustness = 0.20704, runtime = 1.66s
-----------------------------------------------------------------
model = models/mnist_resnet_2, norm = 2, activation = tanh
robustness = 0.19130, runtime = 1.70s
-----------------------------------------------------------------
model = models/mnist_resnet_2, norm = 2, activation = arctan
robustness = 0.11770, runtime = 1.67s
```

图 6: 原方法计算出的残差网络认证下界

```
Converting CNN to MLP
model = models/mnist_cnn_4layer_20_3, numlayer = 4, filters = 20, kernel size = 3
accuracy = 0.986
```

图 7: 将 CNN 转换为 MLP

```
CNN-Cert
model = models/mnist_cnn_4layer_5_3, norm = i, activation = ReLU
robustness = 0.04920, runtime = 0.86s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = i, activation = sigmoid
robustness = 0.06713, runtime = 1.20s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = i, activation = tanh
robustness = 0.02341, runtime = 1.38s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = i, activation = arctan
robustness = 0.02235, runtime = 1.05s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = 2, activation = ReLU
robustness = 0.17936, runtime = 0.88s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = 2, activation = sigmoid
robustness = 0.37226, runtime = 1.25s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = 2, activation = tanh
robustness = 0.15776, runtime = 1.41s
-----------------------------------------------------------------
model = models/mnist_cnn_4layer_5_3, norm = 2, activation = arctan
robustness = 0.14949, runtime = 1.03s
```

图 8: 改进后方法计算出的 CNN 认证下界

```
CNN-Cert
model = models/mnist_resnet_2, norm = i, activation = ReLU
robustness = 0.03961, runtime = 1.61s
----------------------------------------------------------------
model = models/mnist_resnet_2, norm = i, activation = sigmoid
robustness = 0.08052, runtime = 1.97s
----------------------------------------------------------------
model = models/mnist_resnet_2, norm = i, activation = tanh
robustness = 0.07092, runtime = 2.07s
----------------------------------------------------------------
model = models/mnist_resnet_2, norm = i, activation = arctan
robustness = 0.04290, runtime = 1.87s
----------------------------------------------------------------
model = models/mnist_resnet_2, norm = 2, activation = ReLU
robustness = 0.15282, runtime = 1.66s
----------------------------------------------------------------
model = models/mnist_resnet_2, norm = 2, activation = sigmoid
robustness = 0.26129, runtime = 1.99s
----------------------------------------------------------------
model = models/mnist_resnet_2, norm = 2, activation = tanh
robustness = 0.22001, runtime = 2.18s
----------------------------------------------------------------
model = models/mnist_resnet_2, norm = 2, activation = arctan
robustness = 0.13677, runtime = 1.91s
```

图 9: 改进后方法计算出的残差网络认证下界

Network		Certified Bounds								Average Compution Time							
		Original				Impoved				Original				Impoved			
	Lp	ReLU	sigmoid	tanh	arctan	ReLU	sigmoid	tanh	arctan	ReLU	sigmoid	tanh	arctan	ReLU	sigmoid	tanh	arctan
MINST, 4 layer	L∞	0.04061	0.06536	0.02226	0.02178	0.04061	0.06713	0.02341	0.02235	0.91	0.99	1.18	2	0.89	1.2	1.38	1.05
5 filters	L2	0.14535	0.34444	0.15177	0.13913	0.14535	0.37226	0.15776	0.14949	0.9	1	1.32	1.05	0.88	1.25	1.41	1.03
8680 hidden nodes	L1	0.27636	0.76413	0.36415	0.33487	0.27636	0.82089	0.37755	0.36526	0.86	0.96	1.24	1.07	0.88	1.25	1.43	1.07
MINST, ResNet-2	L∞	0.03594	0.06083	0.06046	0.03534	0.03594	0.08052	0.07092	0.0429	1.56	1.65	1.67	1.64	1.65	1.97	2.07	1.87
	L2	0.135	0.20704	0.1913	0.1177	0.135	0.26129	0.22001	0.13677	1.71	1.66	1.7	1.67	1.67	1.99	2.18	1.91
	L1	0.24881	0.3836	0.32262	0.21401	0.24881	0.48683	0.37014	0.24626	1.65	1.71	1.74	1.68	1.64	2.03	2.09	1.91

图 10: CNN 和残差网络上的认证边界和平均运行时间

6 总结与展望

文章提出了一种通用的且高效的验证框架 CNN-Cert 来验证卷积神经网络的鲁棒性。通过在每个构建块上应用线性边界技术，处理各种各样的网络架构，包括卷积、池化、批归一化、残差块以及通用激活函数。实验结果一致验证了 CNN-Cert 在求解更紧密的非平凡认证边界方面的有效性和运行时效率方面优于其他方法。

在本次复现过程中，基本实现了文章中提出的方法，计算出了神经网络的最小扰动下界，并且在卷积神经网络上进行实验。对于只支持全连接神经网络的方法，把卷积神经网络转换成等效的全连接神经网络再对比数据。可以看到在 ReLU 激活函数的全连接神经网络上，计算出的最小扰动下界与 Fast-Lin 相近，关于平均计算时间的实验中，计算速度相比 Fast-Lin 提高数倍。

在未来，还可以对更多的激活函数进行进一步的研究。除了最广泛使用的 ReLU、tanh、sigmoid、arctan 激活函数之外，还可以将 leaky ReLU、ELU 和 softplus 等其他激活函数合并到框架中。此外，与其他现有的基于近似的方法一样，本方法是非完备验证。完备性的缺失意味着没有计算出经过认证的鲁棒性边界并不意味着网络在该边界内不鲁棒。精化是证明或反驳边界内鲁棒性的有用技术，未来可进一步研究对近似方法的细化方法，以进一步收紧鲁棒性下限，而不损失其可扩展性。

参考文献

[1] SZEGEDY C, ZAREMBA W, SUTSKEVER I, et al. Intriguing properties of neural networks[C]//2nd

International Conference on Learning Representations, ICLR 2014. 2014.

[2] CHEN H, ZHANG H, CHEN P Y, et al. Show-and-fool: Crafting adversarial examples for neural image captioning[J]. arXiv preprint arXiv:1712.02051, 2017, 2.

[3] CHENG M, YI J, CHEN P Y, et al. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples[C]//Proceedings of the AAAI Conference on Artificial Intelligence: vol. 34: 04. 2020: 3601-3608.

[4] ZÜGNER D, AKBARNEJAD A, GÜNNEMANN S. Adversarial attacks on neural networks for graph data[C]//Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018: 2847-2856.

[5] CHEN P Y, ZHANG H, SHARMA Y, et al. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models[C]//Proceedings of the 10th ACM workshop on artificial intelligence and security. 2017: 15-26.

[6] ILYAS A, ENGSTROM L, ATHALYE A, et al. Black-box adversarial attacks with limited queries and information[C]//International Conference on Machine Learning. 2018: 2137-2146.

[7] TU C C, TING P, CHEN P Y, et al. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks[C]//Proceedings of the AAAI Conference on Artificial Intelligence: vol. 33: 01. 2019: 742-749.

[8] CHENG M, ZHANG H, HSIEH C J, et al. Query-efficient hard-label black-box attack: An optimization-based approach[C]//International Conference on Learning Representations. 2019.

[9] SINHA A, NAMKOONG H, DUCHI J C. Certifiable Distributional Robustness with Principled Adversarial Training. CoRR, abs/1710.10571[J]. arXiv preprint arXiv:1710.10571, 2017.

[10] WONG E, KOLTER Z. Provable defenses against adversarial examples via the convex outer adversarial polytope[C]//International Conference on Machine Learning. 2018: 5286-5295.

[11] WENG T W, ZHANG H, CHEN P Y, et al. Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach[C]//International Conference on Learning Representations. 2018.

[12] KATZ G, BARRETT C, DILL D L, et al. Reluplex: An efficient SMT solver for verifying deep neural networks[C]//International conference on computer aided verification. 2017: 97-117.

[13] PECK J, ROELS J, GOOSSENS B, et al. Lower bounds on the robustness to adversarial perturbations [J]. Advances in Neural Information Processing Systems, 2017, 30.

[14] HEIN M, ANDRIUSHCHENKO M. Formal guarantees on the robustness of a classifier against adversarial manipulation[J]. Advances in neural information processing systems, 2017, 30.

[15] RAGHUNATHAN A, STEINHARDT J, LIANG P. Certified Defenses against Adversarial Examples [C]//International Conference on Learning Representations. 2018.

[16] WENG L, ZHANG H, CHEN H, et al. Towards fast computation of certified robustness for relu networks [C]//International Conference on Machine Learning. 2018: 5276-5285.

[17] ZHANG H, WENG T W, CHEN P Y, et al. Efficient neural network robustness certification with general activation functions[C]//Proceedings of the 32nd International Conference on Neural Information Processing Systems. 2018: 4944-4953.

[18] DVIJOTHAM K, STANFORTH R, GOWAL S, et al. A Dual Approach to Scalable Verification of Deep Networks.[C]//UAI: vol. 1: 2. 2018: 3.