Parameterized Abstract Interpretation for Transformer Verification

Pei Huang[1*], **Dennis Wei**[2*], **Omri Isac**[3], **Haoze Wu**[4,5], **Min Wu**[1], **Clark Barrett**[1]
[1]Stanford University, Stanford, USA [2]IBM Research, USA [3]Hebrew University of Jerusalem, Israel
[4]Computer Science at Amherst College, USA [5]VMware Research, USA
{huangpei,minwu,barrett}@stanford.edu, dwei@us.ibm.com,
omri.isac@mail.huji.ac.il, hwu@amherst.edu

Abstract

Transformers based on the self-attention mechanism have become foundational models across a wide range of domains, thereby creating an urgent need for effective formal verification techniques to better understand their behavior and ensure safety guarantees. In this paper, we propose two parameterized linear abstract domains for the inner products in the self-attention module, aiming to improve verification precision. The first one constructs symbolic quadratic upper and lower bounds for the product of two scalars, and then derives parameterized affine bounds using tangents. The other one constructs parameterized bounds by interpolating affine bounds proposed in prior work. We evaluate these two parameterization methods and demonstrate that both of them outperform the state-of-the-art approach which is regarded as optimal with respect to a certain mean gap. Experimental results show that, in the context of robustness verification, our approach is able to verify many instances that cannot be verified by existing methods. In the interval analysis, our method achieves tighter results compared to the SOTA, with the strength becoming more pronounced as the network depth increases.

Introduction

Models grounded in the self-attention mechanism have exhibited remarkable performance across a wide range of tasks. Notably, their success in large language models (LLMs) has significantly advanced the practical deployment of deep learning in diverse real-world domains. However, they also suffer from various security and safety issues (Wang et al. 2023; Wei, Haghtalab, and Steinhardt 2023), e.g., instability to input perturbations (Goodfellow, Shlens, and Szegedy 2015). Such issues must be addressed before deep models can be used in safety-critical scenarios such as autonomous driving (Yan et al. 2025) and medical diagnostics (Guo et al. 2024). This highlights the pressing need for formal verification, a rigorous technique grounded in mathematical reasoning, to systematically analyze, understand, and guarantee the reliable behavior of safety-critical systems (Katz et al. 2017).

The focus of this paper is the verification of Transformers via abstract interpretation, which involves over-approximating the behavior of a neural network with abstract domains (e.g., symbolic lower and upper bounds), with the hope that the property of interests still holds in the abstraction. Unlike exact methods, which can always, in theory, determine whether a property holds (i.e., they are sound and complete), abstract interpretation aims to improve scalability at the cost of completeness. In neural network verification, a key factor in determining whether this method can successfully verify a broader range of cases lies in the approximation error of the abstract domain for each component. However, abstract domain construction for Transformers poses greater challenges than for simpler neural architectures. This is because, in addition to the element-wise activation functions, Transformers introduce the attention mechanism—a more complex form of nonlinearity. Cross-nonlinearity in the attention module, referring to multiplication or division between two perturbed variables, does not arise in feed-forward neural networks. Furthermore, unlike the abstract interpretation techniques employed for element-wise activation functions, which use a one-dimensional space, the abstract interpretation domain for the inner product operation in attention involves a high-dimensional space and must account for interactions across multiple positions in the hidden features.

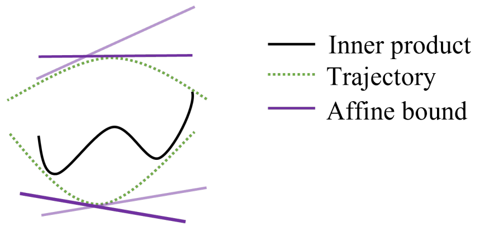

Figure 1: Tunable affine bounds along a Specified trajectory via parameter Control.

To better control the over-approximation error for the attention module and to improve the verification success rate, we propose two *parameterized* symbolic upper and lower bounds for the inner products used in the attention mechanism. With the aid of parameters, task-specific optimization objectives can be introduced to dynamically adjust the abstract domain, thereby improving the verification success rate (Fig 1 provides an intuitive illustration of it.). The first

[*]These authors contributed equally.

approach begins by constructing, for the product xy of two scalars x, y, quadratic upper and lower bounds that are concave and convex, respectively. This ensures that any tangent plane to these quadratic functions serves as a valid affine upper or lower bound for xy. The second approach is based on the interpolation of the two affine bounds proposed in (Shi et al. 2020). We notice that since the mean gap between xy and a bound is a linear function of the bound, any convex combination of bounds that minimize the mean gap is also mean-gap-optimal. Therefore, we parameterize the bounds via interpolation to obtain a family of bounds. Experimental results show that, in the context of robustness verification, our approach is able to verify 85% of the instances that cannot be verified by existing methods. In the interval analysis task, our method achieves tighter results compared to the state of the art, with the advantage becoming more pronounced as the network depth increases. Our contributions are summarized below:

- We propose convex quadratic upper and lower bounds for the inner products employed in attention mechanisms and prove that they minimize the mean gap within this class of bounds. Subsequently, we obtain parameterized affine bounds through the linearization of the quadratic bounds.
- We find that any convex combination of affine bounds proposed in prior work also serves to minimize the mean gap, and parameterize these convex combinations.
- Owing to the flexibility of parameterized bounds, which can be optimized for verification objectives, our methods outperform the SOTA method in both robustness verification and interval analysis tasks.

Related work

Formal DNN verification checks whether a DNN satisfies a property such as the absence of adversarial examples in a given perturbation space. The property is usually depicted by a formal specification, and verifiers aim to provide either a proof of the validity of this property or a counterexample. A range of verification techniques have been developed, mostly for networks with only affine transformations and element-wise activation functions. These techniques are typically categorized into two major classes: exact methods, which guarantee both soundness and completeness, and over-approximate methods, which ensure soundness only. Typical exact methods formalize the verification problem as a Satisfiability Modulo Theories (SMT) problem (Katz et al. 2017; Ehlers 2017; Huang et al. 2017; Jia et al. 2023) or a Mixed Integer Linear Programming (MILP) problem (Cheng, Nührenberg, and Ruess 2017; Fischetti and Jo 2018; Dutta et al. 2018), but their scalability is limited as the problem is NP-hard (Katz et al. 2017). They are thus limited to relatively small networks and are not yet scalable enough to process Transformers at the scale of BERT. Typical over-approximate methods include the linear relaxation of constraint satisfiability problems (Zhou et al. 2024; Wang et al. 2021; Wong and Kolter 2018) and abstract interpretation (Gehr et al. 2018; Singh et al. 2018, 2019). Notably, methods leveraging abstract interpretation scale more effectively in practice. Therefore, in this paper, we focus on the over-approximation approach based on abstract interpretation.

The central task of abstract interpretation–based methods is to construct an abstract domain that over-approximates the nonlinear transformations in neural networks with minimal approximation error. Typically, the abstract domain is required to be a convex set to facilitate the computation of concrete upper and lower bounds. In most cases, affine upper and lower bounds are considered ideal, as they significantly improve verification efficiency and enhance scalability. Shi et al. (Shi et al. 2020) were the first to propose a method for verifying Transformers. They introduced unparameterized affine bounds for the products of scalars that make up the inner products in the attention module. To the best of our knowledge, (Shi et al. 2020) is the state-of-the-art for approximating inner products for Transformer verification. Since this is also our focus, we consider (Shi et al. 2020) as our baseline. Bonaert et al. (Bonaert et al. 2021) proposed multi-norm Zonotope abstract domain for Transformer verification. Although this approach does not outperform the state-of-the-art approach based on backward propagation (Shi et al. 2020) in terms of verification results, it achieves notable improvements in efficiency. In an orthogonal direction, Wei et al. (2023) and Zhang et al. (2024) proposed improved upper and lower bounds on the softmax operation in the attention mechanism. The work of (Wei et al. 2023) is therefore complementary to ours and could be combined with ours in the future. Outside of Transformers, the idea of parameterized affine bounds has been considered in the past. For example, Huang et al. (Zhang et al. 2018) parameterized the affine lower bound of a ReLU connection with a real variable and optimized it with respect to the output bound. In contrast, we propose two novel parameterized bounds for inner products in Transformers.

Preliminaries

The verification problem for most neural networks can be framed as determining whether the output of the network satisfies a given property $\mathcal{P}$ over a specified input space. Thus, the main goal in this work is to compute a provable range of possible output values for every node in every layer.

Assume that the input to the network is denoted by $\mathbf{X}^0$ and the output values of all neurons in the l-th sub-layer is denoted by $\mathbf{X}^l$; formally, we aim to find bounds on $\mathbf{X}^l$ as a function of the input $\mathbf{X}^0$:

$$\underline{W}^{(0,l)}\mathbf{X}^0+\underline{b}^{(0,l)} := L(\mathbf{X}^0) \leq \mathbf{X}^l \leq U(\mathbf{X}^0) := \overline{W}^{(0,l)}\mathbf{X}^0+\overline{b}^{(0,l)}, \tag{1}$$

where $L(\mathbf{X}^0)$ and $U(\mathbf{X}^0)$ denote the symbolic expressions of the lower and upper bounds and parameters $\underline{W}^{(0,l)}$, $\underline{b}^{(0,l)}$, $\overline{W}^{(0,l)}$ and $\overline{b}^{(0,l)}$ are what we aim to compute.

Once the lower and upper bounds L and U have been established for each layer in the neural network, we can obtain the range of possible output values for every node from the extrema of L and U. To obtain the bounds L and U via a backward manner, we first establish how to represent the upper and lower bounds of $\mathbf{X}^l$ in terms of $\mathbf{X}^{l-1}$. Notably, all symbolic bounds are expressed in affine form, as this structure facilitates more efficient and practical computations.

Backward bound propagation (Shi et al. 2020) The process of obtaining L and U can be carried out using a method called backward bound propagation. This approach is analogous to the classical backpropagation used for gradient computation in deep learning frameworks, as it also performs iterative computations on the computational graph in a reverse manner. First, the algorithm constructs upper and lower bounds for each $\mathbf{X}_j^l$ (in practice, for the output of each computational node), which are linear functions of the outputs from the previous layer, so the bounds for $\mathbf{X}^l$ can be expressed in the form:

$$\underline{W}^{(l-1,l)}\mathbf{X}^{l-1} + \underline{b}^{(l-1,l)} \leq \mathbf{X}^l \leq \overline{W}^{(l-1,l)}\mathbf{X}^{l-1} + \overline{b}^{(l-1,l)} \tag{2}$$

The superscript $(l, l-1)$ on the weight matrix and bias indicates that they characterize the relationship between the outputs of layer l and layer $l-1$. One step of backward bound propagation is to compute the weight matrix ($\underline{W}^{(l-2,l)}$, $\overline{W}^{(l-2,l)}$), and bias ($\underline{b}^{(l-2,l)}$, $\overline{b}^{(l-2,l)}$) in equation:

$$\underline{W}^{(l-2,l)}\mathbf{X}^{l-2} + \underline{b}^{(l-2,l)} \leq \mathbf{X}^l \leq \overline{W}^{(l-2,l)}\mathbf{X}^{l-2} + \overline{b}^{(l-2,l)}, \tag{3}$$

Thus, we have

$$\underline{W}^{(l-2,l)} = \underline{W}^{(l-1,l)}_{(+)}\underline{W}^{(l-2,l-1)} + \underline{W}^{(l-1,l)}_{(-)}\overline{W}^{(l-2,l-1)} \tag{4a}$$

$$\overline{W}^{(l-2,l)} = \overline{W}^{(l-1,l)}_{(+)}\overline{W}^{(l-2,l-1)} + \overline{W}^{(l-1,l)}_{(-)}\underline{W}^{(l-2,l-1)} \tag{4b}$$

$$\underline{b}^{(l-2,l)} = \underline{b}^{(l-1,l)} + \underline{W}^{(l-1,l)}_{(+)}\underline{b}^{(l-2,l-1)} + \underline{W}^{(l-1,l)}_{(-)}\overline{b}^{(l-2,l-1)} \tag{4c}$$

$$\overline{b}^{(l-2,l)} = \overline{b}^{(l-1,l)} + \overline{W}^{(l-1,l)}_{(+)}\overline{b}^{(l-2,l-1)} + \overline{W}^{(l-1,l)}_{(-)}\underline{b}^{(l-2,l-1)} \tag{4d}$$

where the subscripts $(+)$ and $(-)$ mean to retain the positive and negative elements in a matrix/vector, respectively, and to set the other elements to 0. The algorithm iteratively performs this process in a backward manner until $\underline{W}^{(0,l)}$, $\overline{W}^{(0,l)}$, $\underline{b}^{(0,l)}$ and $\overline{b}^{(0,l)}$ are obtained. In this context, we only need to concentrate on developing bounds of the form presented in Equation (2) for each layer. The construction of symbolic bounds for various nodes in Transformers (e.g. ReLU, Exp, and Square) has been well established in prior work (Shi et al. 2020). Therefore, we do not elaborate on these and focus solely on constructing the symbolic upper and lower bounds for the inner products within the attention module in form (2).

Inner products in attention Two inner product operations are involved in the computation of the attention module. The output of one self-attention head is given by:

$$B = \frac{Q^T K}{\sqrt{d_K}}, \qquad Z = V \cdot \sigma(B), \tag{5}$$

where $Q = [q_1, q_2, \cdots, q_n]$, $K = [k_1, k_2, \cdots, k_n]$ and $V = [v_1, v_2, \cdots, v_n]$ are the query, key, and value matrices respectively, with $q_i, k_i \in \mathbb{R}^{d_K}$, $v_i \in \mathbb{R}^{d_V}$ and $\sigma : R^{n\times n} \to R_+^{n\times n}$ is the row-wise softmax function applied to matrix B. Both B_{ij} and Z_{ij} are inner products,

$$B_{ij} = \frac{q_i^T k_j}{\sqrt{d_K}} = \frac{1}{\sqrt{d_K}}\sum_{m=1}^{d_K} Q_{mi}K_{mj}, \quad i, j \in [\![1, n]\!] \tag{6a}$$

$$Z_{ij} = \sum_{m=1}^{n} V_{im}\sigma(B)_{mj}, \quad i \in [\![1, d_V]\!],\ j \in [\![1, n]\!]. \tag{6b}$$

In the following sections, we focus on bounding each product xy of scalars x, y that appears in (6).

Motivation for parameterization of bounds

The effectiveness of verifying a property based on over-approximation methods typically depends on the approximation error (or gap) between the approximating function and the original function. Most existing work evaluates the quality of approximations based on the *mean* gap with respect to the *immediately previous layer*, for example the difference between $\mathbf{X}^l$ and its lower bound in (2) averaged over $\mathbf{X}^{l-1}$. However, for specific properties and verification instances, approximations that are optimal in terms of this mean gap do not necessarily yield the best verification results. For example, let $F(X) = [F_0(X), F_1(X)] \in \mathbb{R}^2$ denote the output logits of a Transformer model for a binary classification task. Given an input domain $\mathbb{S}$ and correct label y, the adversarial robustness property can be written as:

$$\forall X \in \mathbb{S} \quad F_y(X) - F_{1-y}(X) > 0 \tag{7}$$

Suppose

$$X^* = \arg\min_{X\in\mathbb{S}} \left(F_y(X) - F_{1-y}(X)\right). \tag{8}$$

We only need to prove that $F_y(X^*) - F_{1-y}(X^*) > 0$ holds in order to establish the validity of formula (7). When using an over-approximation approach, the success of the verification depends on the approximation error of the bounds for $F(X^*)$. In other words, whether the property can be successfully verified is determined not by the mean gap, but by the gap at the point X^*. Moreover, since X is the input and $F(X)$ are the output logits, the bounds on $F(X)$ are backpropagated bounds and not bounds involving adjacent layers. In practice however, it is intractable to directly find X^* and account for all the backpropagation needed to bound $F(X)$, making it difficult to construct optimal bounds.

As an alternative, we aim to design a more flexible bounding mechanism for Transformers that can be optimized during the verification process based on the specific verification task. For example, during adversarial robustness verification, the symbolic bounds at each layer can be adaptively optimized to tighten the lower bound on the output $F_y(X) - F_{1-y}(X)$, with the hope of reducing the approximation error with respect to $F(X^*)$.

Quadratic bounds

In this section, we derive quadratic lower and upper bounds on the product xy of two scalars x and y, where (x, y) can range over the region $[l_x, u_x] \times [l_y, u_y]$. These bounds are to be applied to the inner products in (6). We constrain the lower bound to be convex in (x, y) and the upper bound to be concave, so that tangent planes to these convex/concave functions are also guaranteed to be lower/upper bounds on xy. This will allow us to obtain a parameterized family of bounds. Given these curvature constraints, the bounds are optimal in minimizing the mean difference between the bound and xy, i.e., the mean gap, under the assumption that the inputs are uniformly distributed over the region $[l_x, u_x] \times [l_y, u_y]$. We focus first on the lower bound and then derive an upper bound in by proceeding analogously.

Convex lower bound

Problem statement We denote the lower bound by $L(x, y)$ and define the gap accordingly as $G(x, y) = xy - L(x, y)$. Since $L(x, y)$ is quadratic and constrained to be convex as mentioned above, we parameterize it as follows:

$$L(x,y) = \frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} A_{xx} & A_{xy} \\ A_{xy} & A_{yy} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_x & b_y \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + c, \tag{9}$$

with parameters $A \succeq 0$ (a 2×2 symmetric matrix), $b \in \mathbb{R}^2$, $c \in \mathbb{R}$, and where the positive semidefinite constraint $A \succeq 0$ ensures convexity. Our goal is to minimize the mean gap $\mathbb{E}[G(x, y)]$ with respect to a uniform distribution over the domain, $(x, y) \sim U([l_x, u_x] \times [l_y, u_y])$, subject to the gap being non-negative (i.e., $L(x, y)$ being a valid lower bound) over the domain:

$$\min_{A \succeq 0, b, c} \quad \mathbb{E}[G(x,y)] \tag{10}$$

$$\text{s.t.} \qquad G(x,y) \geq 0 \quad \forall (x,y) \in [l_x, u_x] \times [l_y, u_y]. \tag{11}$$

The following theorem states an optimal lower bound of the form in (9) and the optimal mean gap that it achieves. We define $m_x = (u_x + l_x)/2$ and $\epsilon_x = (u_x - l_x)/2$ as the midpoint and half-width of the interval $[l_x, u_x]$, and similarly m_y, ϵ_y. The proof of Theorem 1 is in the Appendix.

Theorem 1 *The mean gap in bound optimization problem* (10) *is minimized at the following parameter values:* $A_{xx} = \epsilon_y/\epsilon_x$, $A_{yy} = \epsilon_x/\epsilon_y$, $A_{xy} = 1$, $b_x = -\epsilon_y m_x/\epsilon_x$, $b_y = -\epsilon_x m_y/\epsilon_y$, *and* $c = \epsilon_y m_x^2/(2\epsilon_x) + \epsilon_x m_y^2/(2\epsilon_y) - \epsilon_x \epsilon_y$. *The resulting bound (after simplification) is*

$$L(x,y) = xy + \frac{\epsilon_y}{2\epsilon_x}(x - m_x)^2 + \frac{\epsilon_x}{2\epsilon_y}(y - m_y)^2 - \epsilon_x \epsilon_y. \tag{12}$$

The mean gap achieved by (12) *is* $(2/3)\epsilon_x\epsilon_y$.

Concave upper bound

We may now proceed by analogy with section for lower bound to obtain a concave quadratic upper bound $U(x, y)$ that minimizes the mean gap $\mathbb{E}[G(x, y)]$, where now $G(x, y) = U(x, y) - xy$. The key step is to parameterize $U(x, y)$ by the *negative* of the right-hand expression in (9), so that the negation of A and specifically the constraint $-A \preceq 0$ ensure that $U(x, y)$ is concave. The gap $G(x, y) = U(x, y) - xy$ is then given by the same expression as that for the lower bound, but with certain terms negated. The details of this are provided in the Appendix. The rest of the proof of Theorem 1 goes through with these sign changes, yielding the following result.

Corollary 1 *The following upper bound minimizes the mean gap in bound problem* (10)*:*

$$U(x,y) = xy - \frac{\epsilon_y}{2\epsilon_x}(x - m_x)^2 - \frac{\epsilon_x}{2\epsilon_y}(y - m_y)^2 + \epsilon_x \epsilon_y. \tag{13}$$

The mean gap achieved by (13) *is also* $(2/3)\epsilon_x\epsilon_y$.

Two parameterized affine bounds

In this section, we present two forms of parameterized affine bounds. The first one is based on the tangent planes derived from the quadratic bounds given in section for quadratic bounds, whereas the second method relies on an interpolation scheme constructed from affine bounds proposed in prior work. Then, we introduce the method to optimize them.

Tangent planes to quadratic bounds

Equations (12) and (13) provide a convex quadratic lower bound $L(x, y)$ and concave quadratic upper bound $U(x, y)$ on the product of scalars xy over the input region $[l_x, u_x] \times [l_y, u_y]$. Since $L(x, y)$ is convex, any tangent plane to $L(x, y)$ bounds it from below, and is hence also a lower bound on xy. Similarly, any tangent plane to $U(x, y)$ is also an upper bound on xy. A plane tangent to $L(x, y)$ (similarly for $U(x, y)$) at point $(\underline{x_0}, \underline{y_0})$ is given by

$$\begin{aligned} \hat{L}_{\underline{x_0},\underline{y_0}}(x,y) = L(\underline{x_0},\underline{y_0}) &+ \frac{\partial L}{\partial x}(\underline{x_0},\underline{y_0})(x - \underline{x_0}) \\ &+ \frac{\partial L}{\partial y}(\underline{x_0},\underline{y_0})(y - \underline{y_0}), \end{aligned} \tag{14}$$

where $(\partial L/\partial x)(\underline{x_0}, \underline{y_0})$, $(\partial L/\partial y)(\underline{x_0}, \underline{y_0})$ are the partial derivatives of $L(x, y)$ evaluated at $(\underline{x_0}, \underline{y_0})$. For the expressions in (12) and (13), we have

$$\begin{aligned} \hat{L}_{\underline{x_0},\underline{y_0}}(x,y) = L(\underline{x_0},\underline{y_0}) &+ \left(\underline{y_0} + \frac{\epsilon_y}{\epsilon_x}(\underline{x_0} - m_x)\right)(x - \underline{x_0}) \\ &+ \left(\underline{x_0} + \frac{\epsilon_x}{\epsilon_y}(\underline{y_0} - m_y)\right)(y - \underline{y_0}). \end{aligned} \tag{15}$$

Similarly, a tangent plane for $U(x, y)$ at point $(\overline{x_0}, \overline{y_0})$ is given by

$$\begin{aligned} \hat{U}_{\overline{x_0},\overline{y_0}}(x,y) = U(\overline{x_0},\overline{y_0}) &+ \left(\overline{y_0} - \frac{\epsilon_y}{\epsilon_x}(\overline{x_0} - m_x)\right)(x - \overline{x_0}) \\ &+ \left(\overline{x_0} - \frac{\epsilon_x}{\epsilon_y}(\overline{y_0} - m_y)\right)(y - \overline{y_0}). \end{aligned} \tag{16}$$

As an example, choosing the tangent point $(\underline{x_0}, \underline{y_0}) = (\overline{x_0}, \overline{y_0}) = (m_x, m_y)$ yields particularly simple expressions:

$$\hat{L}_{m_x,m_y}(x,y) = m_y x + m_x y - m_x m_y - \epsilon_x \epsilon_y, \quad (17a)$$

$$\hat{U}_{m_x,m_y}(x,y) = m_y x + m_x y - m_x m_y + \epsilon_x \epsilon_y. \quad (17b)$$

It can be verified that the mean gaps $\mathbb{E}[xy - \hat{L}_{m_x,m_y}(x,y)]$, $\mathbb{E}[\hat{U}_{m_x,m_y}(x,y) - xy]$ are both $\epsilon_x\epsilon_y$ for the tangent bounds in (17), compared to $(2/3)\epsilon_x\epsilon_y$ for the quadratic bounds (12), (13).

Interpolation of existing affine bounds

Previous work (Shi et al. 2020) derived affine bounds on the product of scalars xy, also over the input region $[l_x, u_x] \times [l_y, u_y]$. They showed that within the class of affine bounds, their bounds minimize a quantity equivalent to the mean gap over $[l_x, u_x] \times [l_y, u_y]$. In their main paper, the following bounds are presented:

$$\dot{L}(x,y) = l_y x + l_x y - l_x l_y, \quad (18a)$$

$$\dot{U}(x,y) = u_y x + l_x y - l_x u_y, \quad (18b)$$

while in Appendix C, the authors show that the following alternative bounds achieve the same minimal mean gap:

$$\dot{L}(x,y) = u_y x + u_x y - u_x u_y, \quad (19a)$$

$$\dot{U}(x,y) = l_y x + u_x y - u_x l_y. \quad (19b)$$

Our contribution here is to generalize this result by observing that, since the mean gap $\mathbb{E}[xy - \dot{L}(x,y)]$ is a linear function of $\dot{L}(x,y)$, any convex combination of (18a), (19a) also minimizes the mean gap. The same observation holds for upper bounds (18b), (19b). Using $\underline{\alpha}, \overline{\alpha} \in [-1,1]$ to parameterize these convex combinations, we obtain the families of bounds below:

$$\dot{L}_{\underline{\alpha}}(x,y) = (m_y + \underline{\alpha}\epsilon_y)x + (m_x + \underline{\alpha}\epsilon_x)y - m_x m_y - \epsilon_x\epsilon_y - \underline{\alpha}(m_x\epsilon_y + m_y\epsilon_x), \quad (20a)$$

$$\dot{U}_{\overline{\alpha}}(x,y) = (m_y - \overline{\alpha}\epsilon_y)x + (m_x + \overline{\alpha}\epsilon_x)y - m_x m_y + \epsilon_x\epsilon_y + \overline{\alpha}(m_x\epsilon_y - m_y\epsilon_x). \quad (20b)$$

Bounds (18) and (19) correspond respectively to setting $\underline{\alpha} = \overline{\alpha} = -1$ and $\underline{\alpha} = \overline{\alpha} = 1$ in (20). Interestingly, setting $\underline{\alpha} = \overline{\alpha} = 0$ coincides with the tangent bounds in (17), i.e., $\dot{L}_0(x,y) = \hat{L}_{m_x,m_y}(x,y)$ and $\dot{U}_0(x,y) = \hat{U}_{m_x,m_y}(x,y)$.

Optimization of bound parameters

To simplify implementation and enhance optimization smoothness, we reformulate the optimization procedures for both types of bounds into an unconstrained optimization problem. A sigmoid function is used to transform the optimization variables $\underline{x_0}$, $\underline{y_0}$, $\overline{x_0}$, $\overline{y_0}$, $\underline{\alpha}$ and $\overline{\alpha}$ into $\lambda \in (-\infty, +\infty)$, such as:

$$\underline{x_0} = l_x + (u_x - l_x)\frac{1}{1+e^{-\lambda}}, \quad \underline{\alpha} = \frac{2}{1+e^{-\lambda}} - 1. \quad (21)$$

The representations of the remaining variables follow the same pattern and are omitted. Initially, we can set $\lambda = 0$ for the first type of bounds and $\lambda = -4$ for the second, in which case $\underline{x_0} = m_x$ and $\underline{\alpha} \approx -1$. Under this setting, both types of bounds are optimal in terms of the mean gap.

Based on equations (15), (16), (20), and 21, our algorithm constructs parameterized affine bounds in the form (2) for the dot products in each attention layer of the Transformer. Subsequently, backpropagation is performed in reverse from layer l to layer 0 to obtain $\underline{W}^{(0,l)}$, $\underline{b}^{(0,l)}$, $\overline{W}^{(0,l)}$, $\overline{b}^{(0,l)}$. Assume that the set of all optimization variables λ up to and including layer l is denoted by Λ_l, then $L^l(\mathbf{X}^0)$ and $U^l(\mathbf{X}^0)$ are parameterized by Λ_l:

$$L^l(\Lambda_l; \mathbf{X}^0) = \underline{W}^{(0,l)}(\Lambda_l)\mathbf{X}^0 + \underline{b}^{(0,l)}(\Lambda_l), \quad (22)$$

$$U^l_{\Lambda_l}(\Lambda_l; \mathbf{X}^0) = \overline{W}^{(0,l)}(\Lambda_l)\mathbf{X}^0 + \overline{b}^{(0,l)}(\Lambda_l) \quad (23)$$

For many verification problems, the input space of interest can be expressed by $\mathbb{S} = \{\mathbf{X}^0 : \|\mathbf{X}^0 - \mathbf{X}^c\|_p \leq \epsilon\}$, where $\mathbf{X}^c$ is the center of the ℓ_p-ball. The concrete lower bound and upper bound (i.e., the minimum/maximum value of symbolic bounds) of X^l_j:

$$CL^l_j(\Lambda_l) = \underline{W}^{(0,K)}_j(\Lambda_l)\mathbf{X}^c + \underline{b}^{(0,K)}_j(\Lambda_l) - \epsilon \cdot \|\underline{W}^{(0,K)}_j(\Lambda_l)\|_q \quad (24a)$$

$$CU^l_j(\Lambda_l) = \overline{W}^{(0,K)}_j(\Lambda_l)\mathbf{X}^c + \overline{b}^{(0,K)}_j(\Lambda_l) + \epsilon \cdot \|\overline{W}^{(0,K)}_j(\Lambda_l)\|_q \quad (24b)$$

where $1/p + 1/q = 1$, with $p, q \geq 1$. It can be observed from the computation process of backward bound propagation that the concrete lower bound $CL^l_j(\Lambda_l)$ and concrete upper bound $CU^l_j(\Lambda_l)$ are differentiable with respect to Λ_l. Therefore, we can optimize the bounds using gradient-based optimization algorithms according to the objectives defined by the verification task.

To obtain tighter concrete bounds at each layer of the neural network and to better construct the symbolic bounds for subsequent layers, we may define the optimization objective as follows and optimize it for each layer:

$$\min_{\Lambda_l} \sum_j (CU^l_j(\Lambda_l) - CL^l_j(\Lambda_l)) \quad (25)$$

For the adversarial robustness verification problem, suppose $F(\mathbf{X})$ consists of $K-1$ layers in total, and y is the gold label of $\mathbf{X}^c$. We can regard $F_y(X) - F_{1-y}(X)$ as an additional K-th layer. We can thus define the optimization objective as that of maximizing the concrete lower bound (i.e. the minimum value of symbolic lower bounds) of X^K:

$$\max_{\Lambda_K} CL^K(\Lambda_K) \quad (26)$$

If there exists a value in the set of variables Λ_l such that $CL^K(\Lambda_K) > 0$, then property P (stated in 7) is successfully verified.

Experiments

To test the effectiveness of our method, we consider the task of computing the maximum safe perturbation radius for adversarial robustness. We also compare the interval sizes that are over-approximated by each method at different layers of the network. **Baseline** refers to the verifier that is SOTA for

Table 1: Comparison of certified safe perturbation radii under different settings.

Dataset	N	#	Acc.	ℓ_p	Baseline		PBverifierT		PBverifierI	
					Avg	#Win	Avg	#Win	Avg	#Win
Yelp	1	60	91.84	ℓ_1	1.1559	0	1.2294	13	**1.2751**	**42**
				ℓ_2	0.3065	1	0.3179	20	**0.3232**	**31**
				ℓ_∞	0.0262	0	0.0269	15	**0.0275**	**33**
	2	60	91.89	ℓ_1	1.0807	2	1.0779	0	**1.1408**	**58**
				ℓ_2	0.2109	4	0.2108	0	**0.2213**	**56**
				ℓ_∞	0.0164	4	0.0164	0	**0.0173**	**56**
	3	60	92.02	ℓ_1	0.6528	0	0.6836	2	**0.7148**	**58**
				ℓ_2	0.1300	0	0.1356	2	**0.1415**	**58**
				ℓ_∞	0.0103	0	0.0107	2	**0.0112**	**58**
SST	1	50	83.94	ℓ_1	2.8586	**22**	2.6208	0	**2.8683**	21
				ℓ_2	0.4657	7	0.4306	7	**0.4680**	**17**
				ℓ_∞	0.0346	14	0.0320	8	**0.0348**	**15**
	2	50	84.06	ℓ_1	2.0560	0	2.0942	0	**2.1242**	**54**
				ℓ_2	0.3291	0	0.3351	5	**0.3362**	**45**
				ℓ_∞	0.0244	0	0.0246	0	**0.0249**	**39**
	3	50	84.61	ℓ_1	1.5051	0	1.5246	2	**1.5540**	**44**
				ℓ_2	0.2706	0	0.2728	0	**0.2786**	**44**
				ℓ_∞	0.0204	0	0.0205	0	**0.0210**	**44**

bounding inner products, as noted in related work. We refer to our verifier that constructs parameterized bounds based on the tangent plane method as **PBverifierT**, and the one that uses the interpolation method as **PBverifierI**. [1] The parameterized bounds were optimized using the gradient descent algorithm implemented in PyTorch, with a step size of 0.2. The computational environment is described in the supplementary material.

We adopted the same experimental setup as used in (Shi et al. 2020). They considered two sentiment classification tasks, Yelp (Zhang, Zhao, and LeCun 2015) and SST(Socher et al. 2013), and trained BERT-like models on these datasets. Each example is a sentence or a sentence segment labeled with a binary sentiment polarity. We consider N-attention-layer models ($N \leq 3$), with 4 attention heads. The hidden sizes of self-attention layers and feed-forward layers are 256 and 512 respectively. ReLU activations are used for feed-forward layers. Due to the limitations of current verification techniques and constrained GPU resources, we did not evaluate them on larger pre-trained models. Nevertheless, our method is expected to exhibit greater advantages on larger-scale architectures.

Comparison of robustness verification results

We compute maximum safe radius for different models on the two datasets. We record the results on 20 correctly classified random test examples from each dataset. For each example/sentence, we select one of the token positions and perturb the corresponding embedding vector within an ℓ_p ball. The total number of sentence-position pairs (referred to as “instances”) is 60 for Yelp and 50 for SST (please see supplement for details). For each instance, we employ a binary search strategy to query the verifier repeatedly with different radii to determine whether robustness is satisfied. As a result, each instance requires approximately 15 queries on average, which is computationally equivalent to verifying around 750~900 instances under a fixed radius. Table 1 presents the maximum safe radius computed by different methods, averaged over the instances, as well as the number of instances on which each method achieves the maximum value. Specifically, if two methods tie for the best performance on a particular instance, neither receives an increment in the ‘#Win’ count. Fig 2 presents the differences of the verification results of our methods and the baseline across all instances under ℓ_2 norm perturbation. The value of each bar represents the difference in the maximum certified radius on the same instance between our two methods and the baseline. The sample indices on the horizontal axis are ordered based on these differences. Points above the difference $= 0$ line indicate that our method can certify a larger perturbation radius for the corresponding instance. Results under ℓ_1 and ℓ_∞ perturbation are provided in the Appendix.

Our method achieves better performance than the baseline on more than 85% of the cases. These results suggest that the proposed parameterized bounds exhibit superior flexibility over fixed bounds, allowing for task-specific optimization and leading to improved verification performance. In theory, PBverifierI should not perform worse than the baseline on any instance since PBverifierI’s bounds subsume the baseline. However, since the problem of optimizing the bounds is non-convex, the algorithm may sometimes fail to find a better solution.

Since our method involves an optimization process over the bound, it is less efficient than the baseline. In the experiments, our methods take up to 100 seconds to verify one in-

[1] The source code and appendix are provided in https://github.com/huangdiudiu/PBVerification-for-Transformers

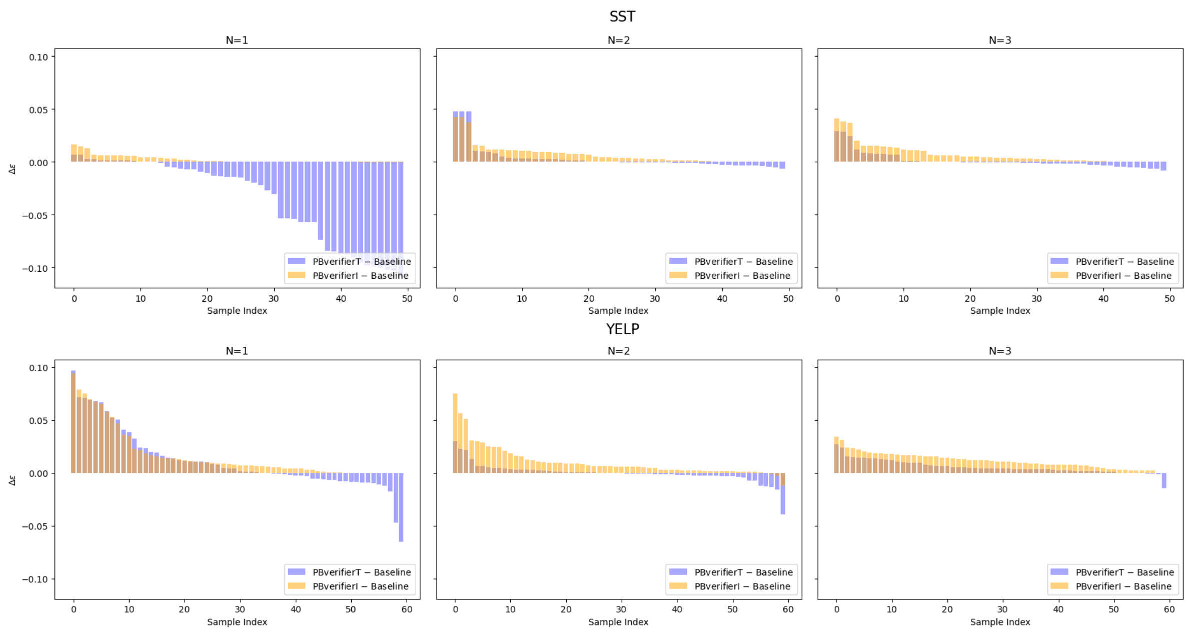

Figure 2: Distribution of maximum robustness radius differences (ℓ_2 perturbation).

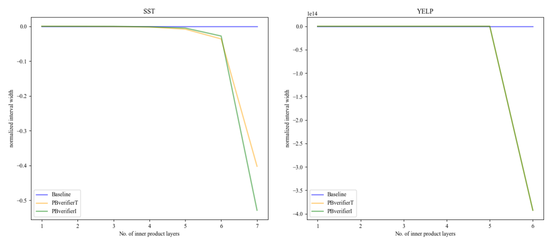

Figure 3: Comparison of normalized interval width among different verifiers at different inner product modules

stance under a fixed radius for a 3-layer networks, whereas the baseline only requires up to 21 seconds. Although our method consumes more computation time than the baseline, the time cost is acceptable when aiming to improve verification precision. As the radius to be verified approaches the real maximum safe radius, the increase in computational difficulty is nonlinear. However, the complexity of our algorithm is a polynomial function of the number of neurons N in the network ($\mathcal{O}(N^2)$).

The results of perturbing two positions at a time are presented in the Appendix. Perturbing more positions (> 3) would significantly reduce the robustness radius, making it difficult to evaluate the effectiveness of different methods, and thus we did not include such experiments. However, for models with two or more attention layers, we observe that perturbing a single token leads to perturbations at all positions starting from the second layer. This demonstrates that our method is capable of handling scenarios where all positions are perturbed.

Tightness of interval analysis

We compare the performance of different methods in interval analysis. Specifically, we report the width of the output intervals, averaged over nodes, for each of the two dot-product modules within every attention layer. The network used consists of 3 attention layers, corresponding to 6 dot-product modules in total. In this experiment, the bound optimization objective is Equation (25). We fix the perturbation radius $\epsilon = 0.3$ under the ℓ_2 norm, which exceeds the average safe radius of the two networks reported in Table 1. A smaller average interval width indicates a tighter result in interval analysis. Figure 3 shows the interval analysis performance across different methods. The data shown in the figure has been normalized using the interval size of the baseline (i.e. the difference from the baseline interval size). The results demonstrate that our bounds can yield tighter interval analysis outcomes after optimization. Moreover, the advantage becomes increasingly pronounced for deeper layers, indicating that parameterized bounds are promising for verifying deeper neural networks.

Conclusion

We propose two parameterized affine bounds on the inner products used in the attention mechanism. The parameterized bounds can be flexibly optimized according to the verification task. Experimental results demonstrate that they can outperform traditional bounds—previously considered optimal in the sense of mean gap—in robustness verification problems and interval analysis. Future research could investigate extending this method to quantized Transformer architectures in order to assess the impact of quantization on the safety of neural networks.

Acknowledgements

This work was funded in part by IBM as a founding member of the Stanford Institute for Human-centered Artificial Intelligence (HAI), a Ford Alliance Project (316280), the National Science Foundation (grant number 2211505), and the Stanford Center for AI Safety.

References

Bonaert, G.; Dimitrov, D. I.; Baader, M.; and Vechev, M. T. 2021. Fast and precise certification of transformers. In Freund, S. N.; and Yahav, E., eds., *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*, 466–481. ACM.

Cheng, C.; Nührenberg, G.; and Ruess, H. 2017. Maximum Resilience of Artificial Neural Networks. In *Automated Technology for Verification and Analysis - 15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings*, volume 10482 of *Lecture Notes in Computer Science*, 251–268. Springer.

Dutta, S.; Jha, S.; Sankaranarayanan, S.; and Tiwari, A. 2018. Output Range Analysis for Deep Feedforward Neural Networks. In *NASA Formal Methods - 10th International Symposium, NFM 2018, Newport News, VA, USA, April 17-19, 2018, Proceedings*, volume 10811 of *Lecture Notes in Computer Science*, 121–138. Springer.

Ehlers, R. 2017. Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks. In *Automated Technology for Verification and Analysis - 15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings*, volume 10482 of *Lecture Notes in Computer Science*, 269–286. Springer.

Fischetti, M.; and Jo, J. 2018. Deep neural networks and mixed integer linear optimization. *Constraints An Int. J.*, 23(3): 296–309.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. T. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, 3–18.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Guo, Y.; Yang, Y.; Zhang, Y.; Wang, Y.; and Wang, Y. 2024. DictLLM: Harnessing Key-Value Data Structures with Large Language Models for Enhanced Medical Diagnostics. In Ku, L.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, 10231–10241. Association for Computational Linguistics.

Huang, X.; Kwiatkowska, M.; Wang, S.; and Wu, M. 2017. Safety Verification of Deep Neural Networks. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, 3–29. Springer.

Jia, F.; Han, R.; Huang, P.; Liu, M.; Ma, F.; and Zhang, J. 2023. Improving Bit-Blasting for Nonlinear Integer Constraints. In Just, R.; and Fraser, G., eds., *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, 14–25. ACM.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, 97–117. Springer.

Shi, Z.; Zhang, H.; Chang, K.-W.; Huang, M.; and Hsieh, C.-J. 2020. Robustness Verification for Transformers. In *International Conference on Learning Representations (ICLR)*.

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. 2018. Fast and effective robustness certification. *Advances in Neural Information Processing Systems*, 31: 10802–10813.

Singh, G.; Gehr, T.; Püschel, M.; and Vechev, M. 2019. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL): 1–30.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1631–1642. ACL.

Wang, B.; Chen, W.; Pei, H.; Xie, C.; Kang, M.; Zhang, C.; Xu, C.; Xiong, Z.; Dutta, R.; Schaeffer, R.; Truong, S. T.; Arora, S.; Mazeika, M.; Hendrycks, D.; Lin, Z.; Cheng, Y.; Koyejo, S.; Song, D.; and Li, B. 2023. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans,LA, USA, December 10 - 16, 2023*.

Wang, S.; Zhang, H.; Xu, K.; Lin, X.; Jana, S.; Hsieh, C.-J.; and Kolter, J. Z. 2021. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*.

Wei, A.; Haghtalab, N.; and Steinhardt, J. 2023. Jailbroken: How Does LLM Safety Training Fail? In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Wei, D.; Wu, H.; Wu, M.; Chen, P.-Y.; Barrett, C.; and Farchi, E. 2023. Convex Bounds on the Softmax Function with Applications to Robustness Verification. In *Proceedings of the 26th International Conference on Artificial Intel-*

ligence and Statistics (AISTATS), volume 206 of *Proceedings of Machine Learning Research*, 6853–6878. PMLR.

Wong, E.; and Kolter, Z. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, 5286–5295.

Yan, T.; Yin, J.; Lang, X.; Yang, R.; Xu, C.; and Shen, J. 2025. OLiDM: Object-aware LiDAR Diffusion Models for Autonomous Driving. In Walsh, T.; Shah, J.; and Kolter, Z., eds., *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, 9121–9129. AAAI Press.

Zhang, H.; Weng, T.-W.; Chen, P.-Y.; Hsieh, C.-J.; and Daniel, L. 2018. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31.

Zhang, X.; Zhao, J. J.; and LeCun, Y. 2015. Character-level Convolutional Networks for Text Classification. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 649–657.

Zhang, Y.; Shen, L.; Guo, S.; and Ji, S. 2024. GaLileo: General Linear Relaxation Framework for Tightening Robustness Certification of Transformers. In Wooldridge, M. J.; Dy, J. G.; and Natarajan, S., eds., *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, 21797–21805. AAAI Press.

Zhou, D.; Brix, C.; Hanasusanto, G. A.; and Zhang, H. 2024. Scalable Neural Network Verification with Branch-and-bound Inferred Cutting Planes. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.