Tighter Truncated Rectangular Prism Approximation for RNN Robustness Verification

Xingqi Lin[1], Liangyu Chen[1]*, Min Wu[1], Min Zhang[1], Zhenbing Zeng[2]

[1]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University
[2]Department of Mathematics, Shanghai University
lychen@sei.ecnu.edu.cn

Abstract

Robustness verification is a promising technique for rigorously proving Recurrent Neural Networks (RNNs) robustly. A key challenge is to over-approximate the nonlinear activation functions with linear constraints, which can transform the verification problem into an efficiently solvable linear programming problem. Existing methods over-approximate the nonlinear parts with linear bounding planes individually, which may cause significant over-estimation and lead to lower verification accuracy. In this paper, in order to tightly enclose the three-dimensional nonlinear surface generated by the Hadamard product, we propose a novel truncated rectangular prism formed by two linear relaxation planes and a refinement-driven method to minimize both its volume and surface area for tighter over-approximation. Based on this approximation, we implement a prototype *DeepPrism* for RNN robustness verification. The experimental results demonstrate that *DeepPrism* has significant improvement compared with the state-of-the-art approaches in various tasks of image classification, speech recognition and sentiment analysis.

Introduction

The widespread application of artificial intelligence has raised growing concerns about its security. This is particularly critical in scenarios with low fault tolerance, such as obstacle detection for autonomous driving and diagnostic classification in medical imaging, where neural network errors can lead to catastrophic consequences. Some studies (Su, Vargas, and Sakurai 2019) even show that one single pixel attack can fool neural networks, exposing the vulnerability to adversarial attacks. Nevertheless, due to the black-box essentiality and vast scale of neural networks, it is not practical to evaluate their security by exhaustively enumerating all possible inputs. Therefore, constructing an effective verification framework to analyze the robustness of these networks is important and necessary.

The problem of neural network robustness verification can be described as follows: Given an input x and a perturbation ϵ, does the result remain consistent with the original x or within an acceptable margin of error? For classification tasks, verification typically involves checking whether the predicted probability of the target class stays higher than that of other classes after perturbation. Current researches have mainly focused on the verification of Feedforward Neural Networks (FNNs), with relatively less attention on Recurrent Neural Networks (RNNs). This restricts the full potentiality of RNNs in vital applications.

RNN is a type of artificial neural network designed to process sequential data. Its architecture maintains a memory of past inputs, making it well-suited for tasks like natural language processing, speech recognition, and time series forecasting. However, their susceptibility to adversarial examples becomes an increasing worry (Papernot et al. 2016). The key challenge in RNN verification is the nonlinearity of the activation and gated functions. Typically, this issue is tackled by over-approximating the initial network to construct a linear problem with relaxed abstract domains. Taking the classic RNN Long Short-Term Memory (LSTM) as an example, nonlinear operations like $\sigma(x) \odot \tanh(y)$, which involve the multiplication of two variables, significantly increase the verification burden. The previous work (Ryou et al. 2021) applies DeepPoly (Singh et al. 2019) to RNN verification, using Linear Programming (LP) to obtain the upper and lower planes of gated functions. The objective function of LP is the sum of the vertical distances between the surface and the planes at sampled points. By minimizing it, one can obtain planes that are closer to the surface. While this method is intuitive, it neglects the relationship between the bounding planes, thus losing verification accuracy.

In this paper, we focus on the relaxation problem of $\sigma(x) \odot \tanh(y)$, analyze the truncated rectangular prism formed by two linear relaxation planes and propose a tighter relaxation method based on the hybrid objective function of its volume and surface area. Through strict mathematical deduction, we demonstrate that the height of the centroid between the upper and lower planes is proportional to the volume, while the surface area is positively correlated with the difference between the maximum value of the upper plane and the minimum value of the lower plane. Therefore, we obtain a more straightforward and theoretical approximation method by using the weighted sum of the height and the difference as the optimization objective. Based on this, we design and implement a RNN verifier called *DeepPrism*, which yields a notable improvement on robustness verification in various tasks of image classification, speech recognition and

*Liangyu Chen is the corresponding author.

sentiment analysis, outperforming previous work.

Our main contributions are as follows:

- We introduce the truncated rectangular prism formed by two relaxation planes and minimize its volume and surface area to achieve a tighter over-approximation, thereby constructing effective abstract domains for RNN robustness verification.
- We propose an over-approximation approach that combines linear programming with a hybrid objective function and abstraction refinement based on different division strategies.
- We implement our approach as a RNN verifier *DeepPrism*, and evaluate it through experiments. The experimental results on four datasets for three tasks show that *DeepPrism* outperforms other SOTA baselines with higher accuracy. The code and data are available in https://github.com/Olinvia/DeepPrism .

We describe our related work, preliminary, methodology, experiments and conclusion in the following sections.

Related Work

Methods for neural network verification can be categorized into exact methods and approximate methods. Exact methods mainly include Satisfiability Modulo Theory (SMT) (Katz et al. 2017, 2019; Duong et al. 2023; Isac et al. 2025) and Mixed Integer Linear Programming (MILP) (Bunel et al. 2018; Dutta et al. 2018; Xue et al. 2022). They can precisely compute the reachable sets of the neural network output but suffer from high complexity, heavy computational cost, and limited scalability. Since around 2018, approximate methods have gradually gained prominence due to their efficiency. Approximate methods include abstract interpretation (Gehr et al. 2018; Singh et al. 2018; Lemesle, Lehmann, and Gall 2024; Marzari, Mastroeni, and Farinelli 2025), symbolic propagation (Wang et al. 2018b,a; Hu et al. 2025), and convex optimization (Müller et al. 2022; Wu et al. 2022), etc. They offer significant advantages in computational efficiency and scalability, making them applicable to larger-scale neural networks and more complex application scenarios.

For RNN verification(Mohammadinejad et al. 2021; Banaajjar et al. 2023), there are three mainstream approximate methods: abstract interpretation, RNN2FNN-based verification, and automata-based methods. Abstract interpretation maps the internal structure of RNN to a set of specific geometric shapes and then verifies whether the abstract domain of the output layer satisfies robustness properties. It is efficient and scalable, but the approximation may lead to a loss of accuracy. RNN2FNN-based verification involves converting the inputs at all time steps into static inputs and applying the verification methods used for FNNs. The methods balance reliability and completeness but come with higher costs. The automata-based method extracts an automaton or finite-state machine from the RNN, which requires a higher level of theoretical knowledge.

Since this paper focuses on the abstract interpretation methods, we summarize them as follows.

(1) Ko et al. (2019), inspired by Fastlin (Weng et al. 2018), which adds linear constraints to neural network operations, first applied abstract interpretation to RNN verification and proposed *POPQORN*. The nonlinear parts are bounded by linear functions, which can be propagated back to the first layer from the output layer recursively. However, this method can handle only a limited number of neurons and may result in overly loose robustness bounds.
(2) Based on *POPQORN*, Du et al .(2021) applied the ideas from DeepZ (Singh et al. 2018) to RNN verification and proposed a tighter verification framework, *Cert-RNN*. Their relaxation strategy for S-shaped activation functions such as sigmoid and tanh is more precise, leveraging the properties of tangents. Additionally, they refined the linear bounds for Hadamard products by conducting a case-by-case analysis, achieving better experimental results than *POPQORN*.
(3) Ryou et al. (2021) proposed a new RNN verifier, *Prover*. They drew on the ideas of DeepPoly and introduced numerical and symbolic bounds for each neuron. For each layer, the bounds are propagated back to the input layer. Moreover, their relaxation approach employs linear programming to find the upper and lower bounding planes individually.
(4) Zhang et al. (2023) proposed *RNN-Guard*, a certified defense against multi-frame attacks for RNNs. They designed an abstract domain called InterZono, which achieves twice the verification precision compared to the Zonotope (Ghorbal, Goubault, and Putot 2009).

Preliminaries

LSTM

LSTM is a type of RNN, whose architecture is shown in Fig. 1. It has three gate functions explained as follows.

The forget gate determines what information should be discarded from the cell state and is represented as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{1}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the activation function, W_f and b_f are the weight and bias, and $[h_{t-1}, x_t]$ represents the concatenation of the previous hidden state and the current input.

The input gate controls what new information should be added to the cell state and decides the extent to which the cell's memory is updated. It can be expressed as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2}$$

$$\tilde{c}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \tag{3}$$

where $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is the activation function, i_t is the output, $\tilde{c}_t$ is the candidate memory cell state, and W_i, W_C and b_i, b_c are the weights and biases, respectively.

The output gate decides what the next hidden state should be based on the current cell state:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{5}$$

$$h_t = o_t \odot \tanh(c_t), \tag{6}$$

where $\odot$ is Hadamard product, o_t is the output, c_t is the cell state, and h_t is the hidden state. The nonlinearity of Eq. 5 and Eq. 6, such as $\odot$ operation, is the challenge of RNN verification.

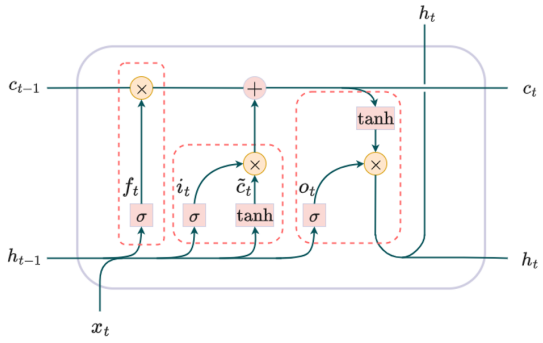

Figure 1: An LSTM cell consists of a forget gate, an input gate, and an output gate. The regions highlighted by the red dashed line indicate the key challenge of over-approximation of $\sigma(x) \odot \tanh(y)$ and $\sigma(x) \odot y$.

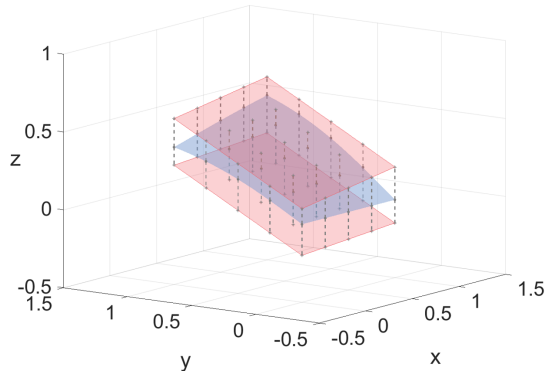

Figure 2: The upper and lower planes computed by linear programming based on the distance sum of sampling points.

DeepPoly

DeepPoly (Singh et al. 2019) is an effective framework for FNN verification. Let the set of neurons at layer ℓ in the network be $X^{(\ell)} = \{x_1^{(\ell)}, x_2^{(\ell)}, \ldots, x_n^{(\ell)}\}$. For each neuron $x_i^{(\ell)}$, there are numerical constraints $l_i \leq x_i^{(\ell)} \leq u_i$ and symbolic constraints $a_i \cdot x_j^{(\ell-1)} + b_i \leq x_i^{(\ell)} \leq a_i' \cdot x_j^{(\ell-1)} + b_i'$, where $l_i, u_i, a_i, b_i, a_i', b_i' \in \mathbb{R}$. Nonlinear neurons at layer ℓ can be linearly over-approximated by neurons at layer $\ell - 1$, and this process can be recursively traced back to the first layer. Therefore, given the input and perturbation, the output range can be calculated.

Tighter Over-approximation

DeepPoly demonstrates that as long as symbolic linear bounds for nonlinear functions can be found, the output range can be determined by using backsubstitution. Thus, the core problem of verification is finding appropriate linear approximation methods. Here, we take $\sigma(x) \odot \tanh(y)$ as an example to discuss how we obtain tighter abstraction domains in abstract interpretation.

Distance-based Method (Ryou et al. 2021)

We first briefly introduce the distance-based method called *Prover* (Ryou et al. 2021). LSTM involves two multiplications that require approximation, and we use $f(x, y) = \sigma(x) \odot \tanh(y)$ as an example. We need to find the upper and lower bounding planes of f such that: $A_l \cdot x + B_l \cdot y + C_l \leq f(x, y) \leq A_u \cdot x + B_u \cdot y + C_u$. It can be transformed as an optimization problem, namely, the variables are the coefficients of the planes A_l, B_l, C_l, A_u, B_u, C_u, the constraints should ensure that the upper bounding plane always lies above the surface and the lower bounding plane always lies below the surface, and the objective function defines how we evaluate the quality of the abstract domain. Ryou et al. proposed minimizing the vertical distance between the surface and the planes, which are expressed as:

$$\begin{aligned} \min_{A_l, B_l, C_l} & \int_{(x,y)\in B} (f(x, y) - (A_l \cdot x + B_l \cdot y + C_l)) \\ \text{s.t. } & A_l \cdot x + B_l \cdot y + C_l \leq f(x, y), \forall (x, y) \in B, \end{aligned} \tag{7}$$

and

$$\begin{aligned} \min_{A_u, B_u, C_u} & \int_{(x,y)\in B} ((A_u \cdot x + B_u \cdot y + C_u) - f(x, y)) \\ \text{s.t. } & A_u \cdot x + B_u \cdot y + C_u \geq f(x, y), \forall (x, y) \in B. \end{aligned} \tag{8}$$

where B is $[l_x, u_x] \times [l_y, u_y]$.

Eq. 7 and Eq. 8 are implemented by sampling, essentially representing the surface features by several points, as shown in Fig. 2. The constraints are satisfied at the n sampled points, and the objective function is the sum of the distances at these n points. The more points sampled, the more precise the approximation, but the time cost increases accordingly. However, sampling cannot fully guarantee soundness, such as ensuring that the lower bounding plane is always beneath the surface. Therefore, after the linear programming is completed and A_l, B_l, and C_l are obtained, it is necessary to check whether the curve surface and the lower plane intersect. If they do, the planes should be adjusted with offsets. The offset algorithm can be referred in (Ryou et al. 2021).

Volume-based Method

The problem in the distance-based method is the independent solving of the upper and lower planes without considering their relationship. One can investigate the truncated rectangular prism formed by four interval constraint planes $x = l_x$, $x = u_x$, $y = l_y$, $y = u_y$ and two linear relaxation planes $A_l \cdot x + B_l \cdot y + C_l$ and $A_u \cdot x + B_u \cdot y + C_u$, as shown in Fig. 3. The curve surface is contained in the prism. Since the four side planes are fixed, an intuitive idea is to make the prism "smaller" for a tighter approximation. Obviously, the metric of volume can be used to measure the approximation of the upper/lower planes and the curve surface.

The volume of prism can be computed as the area of the rectangular base multiplied by the height of the centroid line.

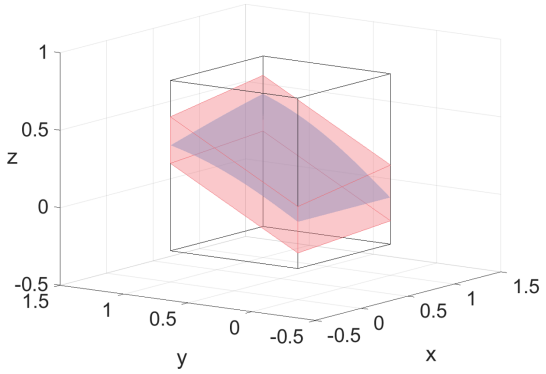

Figure 3: The truncated rectangular prism formed by four interval constraint planes and two relaxation planes.

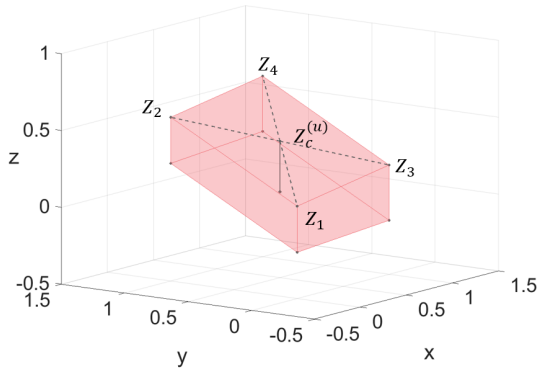

Figure 4: Two important lines of the truncated rectangular prism: The solid line connects the centroids in the upper and lower planes, and the dashed line connects Z_u and Z_l. The solid line "controls" volume, and the dashed line "controls" surface area.

Let $z_c^{(u)} = A_u \cdot \frac{l_x+u_x}{2} + B_u \cdot \frac{l_y+u_y}{2} + C_u$, and $z_c^{(l)} = A_l \cdot \frac{l_x+u_x}{2} + B_l \cdot \frac{l_y+u_y}{2} + C_l$. Thus, the prism's volume V in Fig. 3 can be calculated with

$$\begin{aligned} height &= z_c^{(u)} - z_c^{(l)}, \\ V &= (u_x - l_x) \cdot (u_y - l_y) \cdot height. \end{aligned} \tag{9}$$

It is observed from Eq. 9 that the volume does not depend on the sampling points, thus reducing the influence of sampling randomness. Since l_x, u_x, l_y and u_y are known parameters, the volume-based method is also a linear programming problem:

$$\begin{aligned} &\min_{A_u,B_u,C_u,A_l,B_l,C_l} V \\ &\text{s.t. } A_l \cdot x + B_l \cdot y + C_l \le f(x,y), \forall (x,y) \in B, \\ &\quad A_u \cdot x + B_u \cdot y + C_u \ge f(x,y), \forall (x,y) \in B. \end{aligned} \tag{10}$$

The volume V directly reflects the degree of spatial looseness between the upper and lower planes, and minimizing the volume naturally creates a synergistic relationship between them. Furthermore, the volume-based method requires to solve only one LP problem, while the distance-based method needs two. This can accelerate the calculation.

Hybrid Volume-Area-based Method (*DeepPrism*)

Eq. 9 indicates that the centroid line in Fig. 4 controls the volume, and the plane can be considered as rotating around the centroid. Maintaining roughly the same volume, the goal is to obtain a more "rounded" truncated rectangular prism with a smaller surface area S. In this way, one can obtain a tighter abstract domain in three-dimensional space. The objective function is expressed as:

$$\min_{A_u,B_u,C_u,A_l,B_l,C_l} \alpha \cdot V + (1-\alpha) \cdot S. \tag{11}$$

α weights the contributions of volume and surface area. The surface area S is the sum of the areas of six faces, with four trapezoidal planes on the front, back, left and right only depending on the height of the centroid line. Meanwhile, the areas of the upper and lower planes together are nonlinear, involving the calculation of squares and square roots. The surface area S is positively correlated with the sum of the absolute differences in the z-coordinate between each of the four corner points and the center point, so we can minimize the surface area using this sum. To unify the dimensional scale, we use the centroid height to minimize the volume. Let $\{z_i\}_{i=1}^4$ denote the z-values of the upper plane at the four corner points, computed as $z_i = A_u \cdot x + B_u \cdot y + C_u$ where $(x,y) \in \{l_x, u_x\} \times \{l_y, u_y\}$, and $\{z_i\}_{i=5}^8$ denote the z-values of the lower plane at the four corner points, computed as $z_i = A_l \cdot x + B_l \cdot y + C_l$ where $(x,y) \in \{l_x, u_x\} \times \{l_y, u_y\}$. Then, the objective function in Eq. 11 can be updated to:

$$\begin{aligned} sum &= \sum_{i=1}^{4} \left| z_i - z_c^{(u)} \right| + \sum_{i=5}^{8} \left| z_i - z_c^{(l)} \right|, \\ &\min_{A_u,B_u,C_u,A_l,B_l,C_l} \alpha \cdot height + (1-\alpha) \cdot sum. \end{aligned} \tag{12}$$

Note that the optimization of surface area can be applied to the distance-based method (Ryou et al. 2021). As more points are selected, the average distance becomes closer to the centroid height. Therefore, the distance-area and volume-area methods can be unified, where the latter is more essential and elegant to avoid the sampling limitation.

Verification Process

Our verification process is illustrated in Fig. 5, consisting of four steps: input, approximation, propagation, and output. First, we represent the original sequence data in an interval form with perturbations. Second, the perturbation propagation process is modeled with single-plane and multi-plane approximation methods as explained follows. Third, we use the backsubstitution of DeepPoly to reduce the imprecision. Finally, we obtain the over-approximation of the neural network reachable set.

Single-plane Approximation

The single-plane approximation uses an upper plane and a lower plane to perform linear relaxation of the nonlinear

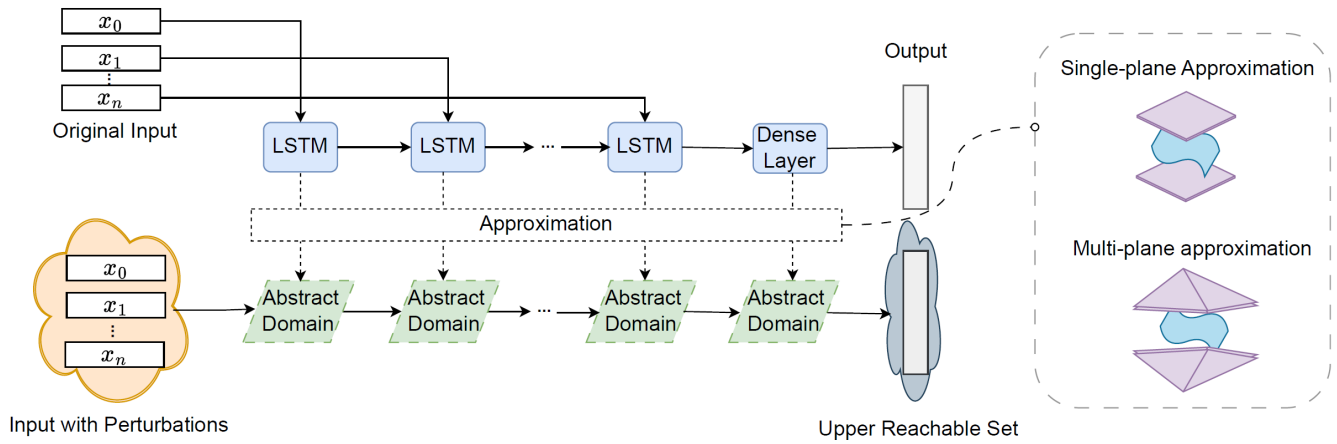

Figure 5: The process of LSTM Certification. The sequence data X go through the LSTM to obtain the output, while the perturbed data $[X-\epsilon, X+\epsilon]$ are processed through abstract domains to compute the reachable set. The abstract domain is executed through either single-plane or multi-plane approximation.

parts, thereby generating the corresponding abstract domain. Through the hybrid volumn-area method, we can obtain the tighter relaxation.

Given an input x with perturbation ϵ, we define a function $g(x, \epsilon, p)$ to represent the gap in predicted probabilities of true label t and prediction class p. Thus, we formalize the robustness verification as $\forall p \neq t, g(x, \epsilon, p) \geq 0$. The single-plane approximation achieves this by solving the LP problem (Eq. 12), obtaining the coefficients of the upper and lower planes, and calculating the value of $g(x, \epsilon, p)$.

In Eq. 12, we balance the influence of volume and surface area by adjusting the weight α. Specifically, α is constrained in $[0, 1]$, and an exhaustive search is performed with a step size of 0.001. For each α, the average value of the objective function $g(x, \epsilon, i)$ is computed, and the α that yields the maximum average value is considered optimal. We conduct experiments and find that the proper value of α is 0.674. This value indicates that giving a slight preference to volume helps to improve the verification accuracy.

Multi-plane Approximation

The single-plane volume-area method only produces a single bound and does not consider global properties of neural networks. Further, this method is, in a sense, greedy: Selecting locally optimal planes for each neuron does not necessarily lead to global optimization. A natural idea is a divide-and-conquer strategy: Divide the LP region and then combine the sub-regions proportionally, with the proportions solved by gradient descent. In this way, the single-plane approximation is upgraded into a multi-plane approximation.

As shown in Fig. 6, we divide the base of the truncated rectangular prism, $[l_x, u_x] \times [l_y, u_y]$ into different parts, such as triangular or rectangular sub-regions. Theoretically, finer divisions improve verification accuracy with higher computational cost, so it is necessary to find a trade-off between accuracy and efficiency.

The sub-regions are denoted as τ_k, where k is the index of the subdivided region, and τ_0 represents the initial region.

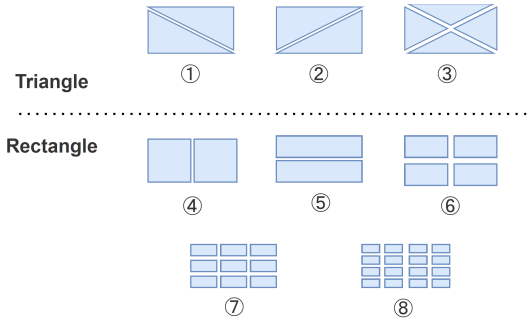

Figure 6: Divisions of multi-plane approximation, which are marked as follows: ① 2-tri-up, ② 2-tri-down, ③ 4-tri, ④ 2-rec-vec, ⑤ 2-rec-hor, ⑥ 4-rec, ⑦ 9-rec, ⑧ 16-rec.

For each sub-region, linear programming is performed to obtain the piecewise upper and lower planes.

Let LB^0 represent the lower plane for the entire region, LB^k represent the lower plane for the sub-region τ_k, and the final plane LB is expressed as a linear combination of LB^k:

$$LB = \sum_{k=0} \lambda_k \cdot LB^k, \sum_{k=0} \lambda_k = 1. \tag{13}$$

Therefore, the robustness verification function $g(x, \epsilon, p)$ is now associated with the weight matrix $\boldsymbol{\lambda} = \{\lambda_0, \lambda_1, \lambda_2, \ldots\}$ and is redefined as $g(x, \epsilon, p, \boldsymbol{\lambda})$. To find $\boldsymbol{\lambda}$, we solve the optimization problem for each prediction label p:

$$\max_{\boldsymbol{\lambda}} g(x, \epsilon, p, \boldsymbol{\lambda}) \geq 0, \tag{14}$$

which can be solved by the gradient descent algorithm in machine learning. Define the loss function as $L = -g(x, \epsilon, p, \boldsymbol{\lambda})$, and update $\boldsymbol{\lambda}$ based on this loss. $L < 0$ indicates that a $\boldsymbol{\lambda}$ has been found that ensures robustness. For specific details, see (Ryou et al. 2021). Using this approach, we refine the abstract domain, allowing us to tighten the output reachable set and improve the verification accuracy.

Name	Task	Input Type	Samples	Classes	Source
MNIST	Image Classification	Image	70,000	10	(Lecun et al. 1998)
Google Speech Commands (GSC)	Speech Recognition	Audio	105,829	35	(Warden 2018)
Free Spoken Digit Dataset (FSDD)	Speech Recognition	Audio	3,000	10	GitHub
Rotten Tomatoes Movie Review (RT)	Sentiment Analysis	Text	43,800	2	(Pang and Lee 2005)

Table 1: Summary of dateset.

Experiments

We evaluate the effectiveness of *DeepPrism* on RNN robustness verification. Specifically, we consider three research questions and answer them respectively.

RQ1: How is the verification accuracy of the single-plane *DeepPrism* compared to *RNN-Guard* and *Prover*?

RQ2: Can *DeepPrism* further improve the verification accuracy in multi-plane approximation?

RQ3: What is the impact of different refinement divisions on the accuracy and running time?

Experimental Setup

Environment. All experiments are executed on a Linux server with the configuration of NVIDIA GeForce 4090, Intel i9-13900K CPU, and 64GB RAM. We use PyTorch 2.4 to implement all models, Gurobi 11.0 as the LP solver.

Dataset. Four datasets corresponding to three tasks are used and listed in Table 1. Specifically, (1) MNIST for image classification; (2) GSC and FSDD for speech recognition; (3) RT for sentiment analysis.

Parameters. We consider three parameters of LSTM: the frame f, the dimension of the hidden state h, and the number of layers ℓ. We use ϵ to denote the perturbation, noting that larger perturbations result in a validation accuracy close to 0, which has no practical significance.

Baselines. Four abstract interpretation-based methods are introduced in the related work. Among four methods , *Prover* achieves the highest precision and largest scale, making it the current state-of-the-art (SOTA) technique. In addition, *RNN-Guard* extends the evaluation to text data and achieves promising results. So, we select *Prover* and *RNN-Guard* as the baselines in the experiments.

Verification of Single-plane Approximation (RQ1)

Image Classification. Fig. 7 shows the verification comparison of three models (*Prover*, *RNN-Guard*, *DeepPrism*) using single-plane approximation. As the perturbation increases, the verification accuracy decreases, with the advantages of *DeepPrism* becoming more evident. *DeepPrism* outperforms other baselines on accuracy with a slight but acceptable increase in computation time.

As to the impact of the model parameters, we set a representative perturbation of 0.012. The performance of three models at different f, h, and ℓ is shown in Table 2. Vertically, an increase in f and ℓ leads to a decline in the accuracy of the verifier, while an increase in h causes an improvement. Horizontally, *DeepPrism* outperforms other baselines under all configurations.

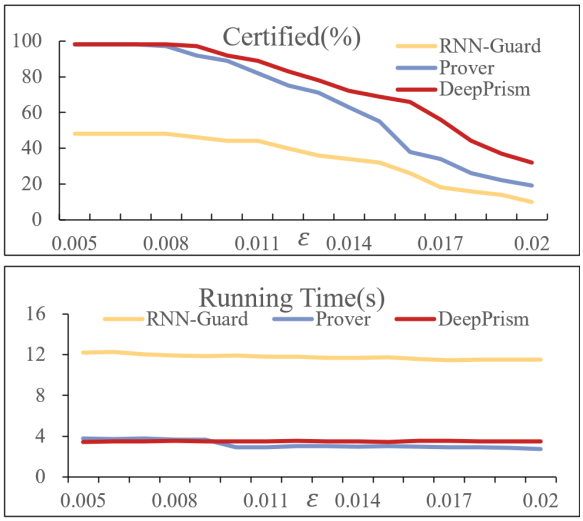

Figure 7: Results on MNIST with different perturbation and models where $f = 4, h = 32$ and $\ell = 2$.

f	h	ℓ	*RNN-Guard* Acc.	*RNN-Guard* Time	*Prover* Acc.	*Prover* Time	*DeepPrism* Acc.	*DeepPrism* Time
4	32	1	40	5.84	45	1.47	**83**	1.79
4	32	2	25	11.83	49	3.02	**68**	3.57
4	32	3	20	17.71	25	4.56	**59**	5.25
4	64	1	53	11.60	81	2.84	**88**	3.15
4	128	1	62	22.84	86	5.46	**93**	6.18
7	32	1	7	10.89	14	2.70	**53**	3.03

Table 2: Comparison of three models with different neural network on MNIST under the perturbation $\epsilon = 0.012$.

Speech Recognition. We compare three baseline methods on the GSC and FSDD datasets (Table 4 and 5). *DeepPrism* demonstrates superior performance by achieving the highest accuracy while simultaneously maintaining the shortest runtime, reflecting an optimal balance between effectiveness and computational efficiency.

Sentiment Analysis. We compare three baseline methods on the RT dataset (Table 6). *DeepPrism* achieves the highest accuracy, indicating its efficacy in sentiment classification.

Verification of Multi-plane Approximation (RQ2)

Theoretically, multi-plane approximation should outperform single-plane approximation. The experimental results in Fig. 8 confirm this.

	① 2-tri-up		② 2-tri-down		③ 4-tri		④ 2-rec-vec		⑤ 2-rec-hor		⑥ 4-rec		⑦ 9-rec		⑧ 16-rec	
ϵ	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time
0.005	**97**	7.79	**97**	7.80	**97**	14.58	**97**	10.81	**97**	8.42	**97**	18.69	**97**	28.18	**97**	38.02
0.008	94	8.82	94	9.14	94	15.97	93	11.17	94	10.93	95	18.98	**97**	28.40	**97**	37.85
0.011	88	9.84	87	9.55	90	17.56	81	12.42	95	11.55	**97**	19.00	**97**	29.89	**97**	37.85
0.014	68	12.13	72	13.68	72	16.87	54	19.62	86	17.26	93	19.74	95	33.09	**96**	38.69
0.017	44	13.08	39	16.88	45	19.86	23	19.17	64	18.98	65	21.42	71	33.15	**76**	37.64
0.020	18	14.47	18	21.91	20	22.07	7	17.06	28	19.53	54	21.72	57	31.68	**64**	38.39

Table 3: Verification accuracy of different divisons on MNIST under different perturbations where $f = 4, h = 32$ and $\ell = 2$.

ϵ(dB)	*RNN-Guard*		*Prover*		*DeepPrism*	
	Acc.	Time	Acc.	Time	Acc.	Time
-100	20	57.12	44	18.06	**46**	15.06
-95	6	59.27	28	17.80	**29**	15.14
-90	0	-	**14**	17.75	**14**	15.03
-85	0	-	6	17.61	**8**	14.97
-80	0	-	**3**	17.63	**3**	14.98

Table 4: Comparison of three models on GSC dataset.

ϵ(dB)	*RNN-Guard*		*Prover*		*DeepPrism*	
	Acc.	Time	Acc.	Time	Acc.	Time
-100	51	68.32	**97**	21.96	**97**	20.99
-95	39	90.35	**90**	21.93	**90**	21.17
-90	27	94.81	86	21.74	**88**	20.89
-85	11	111.98	81	21.80	**84**	20.89
-80	0	-	70	21.78	**72**	21.00

Table 5: Comparsion of three models on FSDD dataset.

At the same time, we observe that the impact of different approximations on the multi-plane approximation method is not significant. *DeepPrism* performs slightly better but with more time consumption, followed by *Prover*. This is because the division process dilutes the impact of linear programming, achieving similar results with more iterations.

Effect of Refinement Divisions (RQ3)

Finally, we investigate the impact of different divisions on multi-plane approximation. Table 3 presents the experimental results under different perturbations. For ① to ⑥, under low perturbations, all models perform similarly under all divisions, where division ③ 4-tri and ⑥ 4-rec slightly outperform others. Under high perturbations, division ⑥ 4-rec clearly outperforms others, maintaining higher accuracy. One can observe that rectangular division provides more uniform coverage of the region, reducing the boundary effects.

In addition, we also test more finer divisions, with the results shown in Division ④, ⑥, ⑦ and ⑧. Finer divisions can better capture surface features, thereby improving overall approximation performance. In the cases of ⑦ 9-rec and ⑧ 16-rec divisions, the verification accuracy has a significant improvement compared to that of ⑥ 4-rec division. Note that more divisions may have a computational burden.

ϵ	*RNN-Guard*		*Prover*		*DeepPrism*	
	Acc.	Time	Acc.	Time	Acc.	Time
0.05	51	57.12	77	22.15	**84**	22.84
0.07	39	58.85	48	23.31	**59**	23.04
0.09	27	59.27	27	25.32	**28**	24.87
0.11	11	58.98	11	22.46	**16**	22.74
0.13	0	-	10	25.43	**12**	22.08
0.15	0	-	0	26.32	**2**	23.90

Table 6: Comparison of three models on RT dataset.

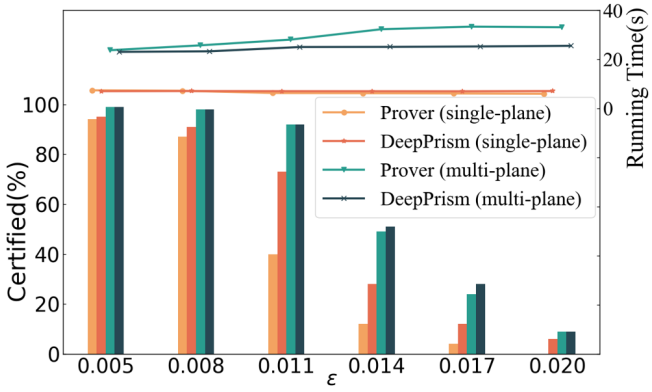

Figure 8: Comparison of two verification methods (single-plane, multi-plane) on two models (*Prover* and *DeepPrism*) evaluated on MNIST using the LSTM model with $f = 4$, $h = 32$ and $\ell = 3$.

Conclusion

We introduce a novel over-approximation method based on the truncated rectangular prism, supported by theoretical guarantees. This method can be effectively applied to RNN verification within the framework of abstract interpretation, leading to significant improvements in experimental results. This work not only improves existing robustness verification techniques but also offers fresh insights into nonlinear analysis in abstract interpretation. Future work will focus on reducing computational overhead of the approach.

Acknowledgements

This work is supported by the National Key Research Project of China (No. 2023YFA1009402), NSFC (Nos. 62272416, 62372176), and Huawei.

References

Baninajjar, A.; Hosseini, K.; Rezine, A.; and Aminifar, A. 2023. SafeDeep: A Scalable Robustness Verification Framework for Deep Neural Networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5.

Bunel, R. R.; Turkaslan, I.; Torr, P.; Kohli, P.; and Mudigonda, P. K. 2018. A unified view of piecewise linear neural network verification. In *Proceedings of the Advances in Neural Information Processing Systems*, 4795–4804.

Du, T.; Ji, S.; Shen, L.; Zhang, Y.; Li, J.; Shi, J.; Fang, C.; Yin, J.; Beyah, R.; and Wang, T. 2021. Cert-RNN: Towards certifying the robustness of recurrent neural networks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 516–534.

Duong, H.; Li, L.; Nguyen, T.; and Dwyer, M. B. 2023. A DPLL(T) framework for verifying deep neural networks. *arXiv preprint arXiv:2307.10266*.

Dutta, S.; Jha, S.; Sankaranarayanan, S.; and Tiwari, A. 2018. Output range analysis for deep feedforward neural networks. In *Proceedings of the NASA Formal Methods Symposium*, 121–138.

Gehr, T.; Mirman, M.; Drachsler-Cohen, D.; Tsankov, P.; Chaudhuri, S.; and Vechev, M. 2018. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, 3–18.

Ghorbal, K.; Goubault, E.; and Putot, S. 2009. The zonotope abstract domain Taylor1+. In *Proceedings of the 21st International Conference of Computer Aided Verification*, 627–633.

Hu, H.; Yang, Y.; Wei, T.; and Liu, C. 2025. Verification of neural control barrier functions with symbolic derivative bounds propagation. In *Proceedings of the 8th Conference on Robot Learning*, 1797–1814.

Isac, O.; Refaeli, I.; Wu, H.; Barrett, C.; and Katz, G. 2025. Proof-Driven clause learning in neural network verification. *arXiv preprint arXiv:2503.12083*.

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 *Proceedings of the 29th International Conference of Computer Aided Verification*, 97–117.

Katz, G.; Huang, D. A.; Ibeling, D.; Julian, K.; Lazarus, C.; Lim, R.; Shah, P.; Thakoor, S.; Wu, H.; Zeljić, A.; et al. 2019. The Marabou framework for verification and analysis of deep neural networks. In *Proceedings of the 31st International Conference of Computer Aided Verification*, 443–452.

Ko, C.-Y.; Lyu, Z.; Weng, L.; Daniel, L.; Wong, N.; and Lin, D. 2019. POPQORN: Quantifying robustness of recurrent neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 3468–3477.

Lecun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Lemesle, A.; Lehmann, J.; and Gall, T. L. 2024. Neural network verification with PyRAT. *arXiv preprint arXiv:2410.23903*.

Marzari, L.; Mastroeni, I.; and Farinelli, A. 2025. Advancing neural network verification through hierarchical safety abstract interpretation. *arXiv preprint arXiv:2505.05235*.

Mohammadinejad, S.; Paulsen, B.; Deshmukh, J. V.; and Wang, C. 2021. DiffRNN: Differential Verification of Recurrent Neural Networks. In *Proceedings of the Formal Modeling and Analysis of Timed Systems: 19th International Conference*, 117–134.

Müller, M. N.; Makarchuk, G.; Singh, G.; Püschel, M.; and Vechev, M. 2022. PRIMA: general and precise neural network certification via scalable convex hull approximations. *Proceedings of the ACM on Programming Languages*, 6: 1–33.

Pang, B.; and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 115–124.

Papernot, N.; McDaniel, P.; Swami, A.; and Harang, R. 2016. Crafting adversarial input sequences for recurrent neural networks. In *Proceedings of the 2016 IEEE Military Communications Conference*, 49–54.

Ryou, W.; Chen, J.; Balunovic, M.; Singh, G.; Dan, A.; and Vechev, M. 2021. Scalable polyhedral verification of recurrent neural networks. In *Proceedings of the 33rd International Conference of Computer Aided Verification*, 225–248.

Singh, G.; Gehr, T.; Mirman, M.; Püschel, M.; and Vechev, M. 2018. Fast and effective robustness certification. In *Proceedings of the Advances in Neural Information Processing Systems*, 10825–10836.

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: 1–30.

Su, J.; Vargas, D. V.; and Sakurai, K. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5): 828–841.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018a. Efficient formal safety analysis of neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, 6369–6379.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018b. Formal security analysis of neural networks using symbolic intervals. In *Proceedings of the 27th USENIX Security Symposium*, 1599–1614.

Warden, P. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

Weng, L.; Zhang, H.; Chen, H.; Song, Z.; Hsieh, C.-J.; Daniel, L.; Boning, D.; and Dhillon, I. 2018. Towards fast computation of certified robustness for Relu networks. In *Proceedings of the International Conference on Machine Learning*, 5276–5285.

Wu, H.; Zeljić, A.; Katz, G.; and Barrett, C. 2022. Efficient neural network analysis with sum-of-infeasibilities. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 143–163.

Xue, H.; Zeng, X.; Lin, W.; Yang, Z.; Peng, C.; and Zeng, Z. 2022. An RNN-based framework for the MILP problem in robustness verification of neural networks. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 1842–1857.

Zhang, Y.; Du, T.; Ji, S.; Tang, P.; and Guo, S. 2023. RNN-Guard: Certified robustness against multi-frame attacks for recurrent neural networks. *arXiv preprint arXiv:2304.07980*.