REACHABILITY ANALYSIS ON RECURRENT NEURAL NETWORKS

Chi Zhang
Department of Computer Science
University of Exeter, UK
{cz338}@exeter.ac.uk

Fu Wang
Department of Computer Science
Guilin University of Electronic & Technology, University of Exeter
{fuu.wanng}@gmail.com

Peipei Xu
Department of Computer Science
University of Liverpool, UK
{peipei.xu}@liverpool.ac.uk

Wenjie Ruan
Department of Computer Science
University of Exeter, UK
{w.ruan}@exeter.ac.uk

ABSTRACT

Verification of neural network plays an essential role in the safety analysis of neural networks. However, most of the recent works focus on the feed-forward neural network. In contrast, verification of the Recurrent Neural Networks (RNNs) is still a relatively young research area. Possible reasons are the special structure components of RNNs, such as gates and loops in LSTM and GRU, causing complexity and strong nonlinearity of the neural networks. We here employ a global optimization based method, RNNgo, to calculate the reachability of RNNs. Under the assumption of Lipschitz continuity, this technique can verify the neural network without the knowledge of inner detail structure. We first prove the Lipschitz continuity of the recurrent layers and then demonstrate RNNgo's capability and efficiency in handling a wide range of RNNs. Through the reachability analysis, RNNgo also tackles the maximum safe radius problem, computing the minimum distance to an adversarial example for a given input example, concreting ground-truth adversarial examples.

1 INTRODUCTION

Deep neural networks (DNNs) or systems with DNN components are widely used in many application scenarios, ranging from image processing to speech recognition (Guo et al., 2018; Graves et al., 2013), from medical diagnose system to self-driving cars (Finlayson et al., 2018; Hecker et al., 2018). In recent years, safety and robustness of DNNs have gained increasing attention, because DNNs are vulnerable to adversarial attacks (Szegedy et al., 2013). Slight and even imperceptible perturbation for inputs can cause significant changes in DNNs' outputs. This kind of misbehavior is described as adversarial examples, which can result in failure of even fatal accidents in safety critical situation (Huang et al., 2020). Therefore it is of vital importance to analyze the safety and robustness of DNNs before employing them in safety-critical applications.

The studies of robustness evaluation of DNNs mainly fall into two categories, adversarial attack approaches and verification-based approaches. The attacking approaches aim to create batches of adversarial examples, which cause the DNNs to make wrong predictions. Representative algorithms for adversarial attack are FGSM (Goodfellow et al., 2014), BIM (Kurakin et al., 2016), the Projected Gradient Descent (PGD) attack (Madry et al., 2017) and Carlini & Wagner (C&W) attacks (Carlini & Wagner, 2017). However, most of the adversarial attacks cannot provide theoretical guarantees on their results, e.g., an unsuccessful attack cannot exclude the existence of adversarial examples and

guarantee the safety of neural networks. Different from the attacking techniques, verification methods check whether a specified property holds on tested neural networks, providing results with provable guarantees. Furthermore, based on their problem formulations and basic algorithms, the verification methods fall into three categories (Huang et al., 2020), i.e., the reachability analysis (Xiang et al., 2017; Gehr et al., 2018; Singh et al., 2018), optimization-based approaches (Tjeng et al., 2017; Bastani et al., 2016; Ruan et al., 2018), and methods combined with searching strategy (Dutta et al., 2017; Katz et al., 2017). However, most of these verification methods only deal with the Feed Forward Neural Networks (FFNNs) and can hardly be generalized on Recurrent Neural Networks (RNNs).

The difficulty of RNN verification lies in the special loop structure and sequential inputs. Existed methods tend to unroll RNN and verify the equivalent FFNNs (Ko et al., 2019). However, unrolling applies only for simple RNN with short sequential input. Long sequence input suffers from the size explosion of the equivalent FFNNs. Furthermore, complex structures such as LSTM and GRU also bring difficulties to unrolling. The concept invariant is brought up in (Jacoby et al., 2020; Zhang et al., 2020) to circumvent the unrolling. However, its application is limited and leads to over-approximation.

This paper proposes a method based on global optimization without suffering the above weakness. Inspired by the optimization research (Gergel et al., 2016; Grishagin et al., 2018), we develop a verification tool RNNgo, which applies for RNNs with various structures, as long as the network is Lipschitz continuous. As far as we know, it is the first verification approach on RNNs without unrolling or over-approximation. Our contributions are summarized as below:

- This paper proposes an unified framework for verification on RNNs, allowing for verification without the knowledge of inner detail structure of RNNs;
- We theoretically prove that the Lipschitz continuity holds on any FFNN and RNN structures and implement the maximal safe radius based on global optimization;
- Our experiments show that RNNgo outperforms the state-of-the-art approach in terms of accuracy and efficiency when dealing with complex model architectures.

2 Related Work

In this part, we discuss several threads of techniques concerning safety analysis. The majority of adversarial attack and verification methods of RNNs make use of the existed methods for FFNNs. The studies on the RNN safety problem are still quite limited.

Adversarial Examples in RNNs Because of the complex structure and sequence input, the method of cultivating the adversarial examples varies. (Papernot et al., 2016) introduced the concept of adversarial example and adversarial sequence. FGSM and forward derivative were used to concrete adversarial examples for LSTM. Instead of analyzing the specific acoustic features, (Gong & Poellabauer, 2017) is the first approach to analyze and perturb the raw waveform of audio directly. It can generate adversarial examples to attack the speech recognition model with RNN and LSTM. Besides, some of the algorithms of white-box attack in FFNN is also applicable for RNNs. (Carlini & Wagner, 2018) developed attack implantation in Deepspeech based on the C&W method (Carlini & Wagner, 2017). Because RNN models are employed in most audio processing systems, there are several specialties of the adversarial attack. For example, the adversarial examples in (Carlini & Wagner, 2018) would lose the adversarial label after adding point-wise noise. Moreover, generating the adversarial example for RNN is time-consuming. (Liu et al., 2019) proposed a weighted-perturbation technology to generate an adversarial example with high efficiency and robustness.

Verification of RNNs (Ko et al., 2019) proposed a new algorithm to quantify the robustness of RNNs, namely Propagatedoutput Quantified Robustness for RNNs (POPQORN). In practice, it introduces the upper and lower planes to bound the non-linear parts of the estimated models. Some verification approaches for FFNN can also be adopted under the RNN scenarios. This kind of methods starts with unrolling of RNNs and use the equivalent FFNNs for further research. In (Zhang et al., 2020), Mixed-integer linear programming (MILP) and polytope propagation are adopted to verify RNN, while (Vengertsev & Sherman, 2020) used a Monte Carlo based approach on RNNs. Both of

them used popular techniques in the verification of FFNNs. However, despite abundantly available techniques for FFNNs, such methods suffer from the complexity of the unrolled RNNs. The size of the equivalent FFNN is quite large, which can lower the efficiency of the verification. (Jacoby et al., 2020) introduced over-approximation and invariant inference, transferring the RNN problem to a simple FFNN model. It is independent of the input size and shows better scalability.

3 Lipschitz Continuity on Recurrent Neural Networks

The Lipschitz continuity is defined in lemma 1. Lipschitz continuity of FFNN has been demonstrated in many researches. Szegedy et al. (Szegedy et al., 2013) proved that deep neural networks with half-rectified layers including convolutional or fully connected layers with ReLU activation functions, max-pooling and contrast-normalization layers are Lipschitz continuous. Later, Ruan et al. (Ruan et al., 2018) showed that a more board of layers are Lipschitz continuous including Softmax layer, Sigmoid and Hyperbolic Tangent activation functions that are widely adopted in feed-forward neural networks. Based on the Lipschitz continuity of FFNN layers, we can demonstrate the Lipschitz continuity of RNNs by transforming them to networks with normal FFNN layers.

Lemma 1. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$, if $\|\partial f(x)/\partial x\| \leq K$ for all $x \in [a, b]^n$, then f is Lipschitz continuous on $[a, b]^n$ and K is its Lipschitz constant, where $\| * \|$ represents a norm operator.*

RNNs can be unfolded into feed-forward neural networks by eliminating loops (Goodfellow et al., 2016). We prove that any RNN with a finite length input is Lipschitz continuous. The basic idea lies on that, due to the finite input, we can unfold an RNN by cloning every node in the RNN along the time sequence, representing the corresponding node in the original RNN at a particular time step. However, there are structures in the unfolded network that may not appear in the normal FFNNs. They are time delay between nodes in Figure 1 and different activation functions in same layer in Figure 3.

Figure 1 shows a time delay from node x_2 to node s_2. In this case we add dummy nodes in the intermediate layers. These dummy nodes use the identity matrix for a weight and use the identity function as an activation function. The generated neural network is equivalent to the direct unrolled RNN and also fulfills the regular FFNN pattern, indicating its Lipschitz continuity. Another special structure, different activation functions in one layer as shown in Appendix Figure 3, usually appears in unfolding RNN with complex hidden layers. This structure is also Lipschitz continuous. The detailed demonstration is in Appendix.

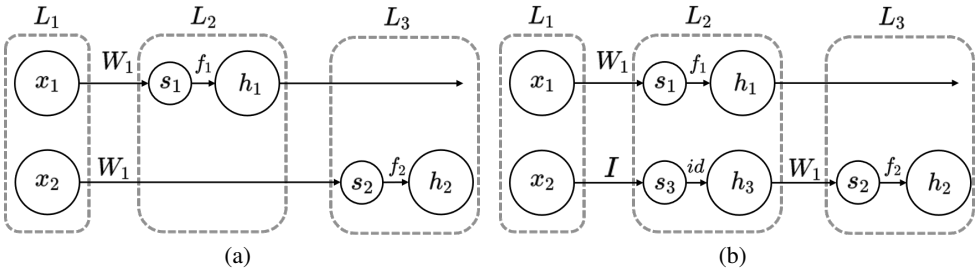

Figure 1: Adding dummy nodes to intermediary layer.

Thus, we have proved that the special structures of unfolded RNNs are also Lipschiz continuity. Based on the Lipschitz continuity of regular FFNN, we can conclude that RNNs are Lipschitz continuous. Therefore, we can adopt the Lipschitz optimization in (Gergel et al., 2016; Grishagin et al., 2018; Ruan et al., 2018) to estimate the output range of RNNs. We use zigzag lines to approach the bound l and further more use binary search to calculate the maximal safe radius. Details can be found in appendix.

Table 1: Average safe radius and standard deviations $(\cdot/\cdot)$ of RNNgo and POPQORN on MNIST.

Models	RNNgo	POPQORN
rnn 7_64	0.7379/0.3312	0.0206/0.017
rnn 4_32	0.8272/0.2996	0.0175/0.0194
lstm 4_32	0.6258/0.5584	0.0075/0.0048
lstm 7_64	0.7215/0.3512	0.0180/0.0150

Table 2: Averaged safe radius and time cost of RNNgo and POPQORN on MNIST.

Models	RNNgo		POPQORN	
	radius	time	radius	time
rnn 4_64	0.1336	253.64s	0.0328	1.31s
rnn 14_64	0.3248	228.23s	0.2344	11.73s
rnn 28_64	0.3551	285.35s	nan	nan
rnn 56_64	0.4369	314.1s	nan	nan
lstm 4_64	0.3195	250.99s	0.0004	307.93s
lstm 14_64	0.3883	382s	0.0123	400.83s
lstm 28_64	0.6469	512.45s	0.0296	532.47s
lstm 56_64	0.6344	491.64s	0.0309	557.22s

4 EXPERIMENTS

In this section, we show the experimental evidence for the utility of our algorithm. Our method is implemented in Python 3.7, running on a computer with i7-4770 CPU, 24GB RAM and NVIDIA Quadro P5000 GPU.

4.1 COMPARISON WITH BASELINE METHOD

We choose POPQORN (Ko et al., 2019) as the baseline method to make a fair comparison because it is easy to re-implement and has a similar verification procedure with our method, i.e., calculating the output range and searching for safe input bounds. The results are summarised in Table 1 2. We carry out the experiment on the MNIST with respect to l_∞, perturbing eight pixels of the input. The maximal safe radius calculated by RNNgo is remarkably larger than POPQORN, in which the over-approximation exists. RNNgo, on the other hand, can calculate the exact output range through global optimization, and further calculate the maximal safe radius. In a word, RNNgo can provide the exact maximal safe radius as long as the RNN is Lipschitz continuous. Because of the black box property and efficiency in analyzing complex models, RNNgo shows a stronger application potential in practical networks than the existing method.

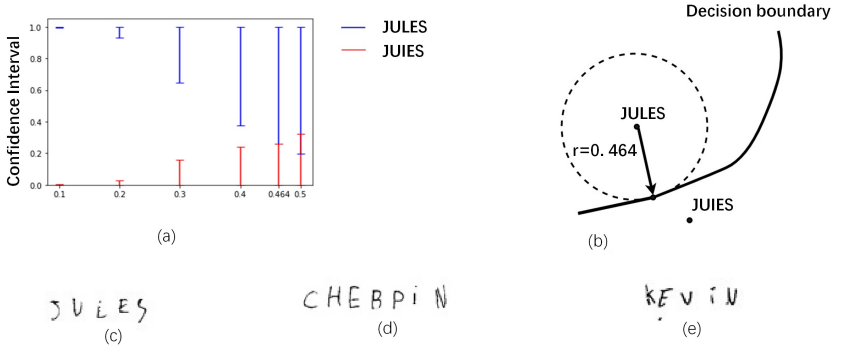

Figure 2: (a) Confidence intervals of label "JUIES" and "JULES" increase with perturbation radius. The intervals get an overlap when $\theta > 0.464$.(b)The maximal safe radius of the original input "JULES" is 0.464. (c) $\theta = 0.464$, confidence for the third letter recognized as "L" 25.96%, for target label "I" 25.97%, recognized as "JUIES"(d) $\theta = 0.638$, confidence for first letter "K" 48.174%, for first letter "L" 48.396%, recognized as "IKEVIN"; (e) $\theta = 0.0631$, confidence for fourth letter recognized as "R" 49.10%, for target label "B" 49.12%, recognized as "CHEBPIN" .

4.2 GROUND-TRUTH ADVERSARIAL EXAMPLES OF CRNN FOR HANDWRITING RECOGNITION

We verify a CRNN with CTC loss for handwriting recognition, with which characters on scanned documents are converted into digital forms. We use RNNgo to calculate the output range and the binary search to find the maximal safe input range, i.e. maximal safe radius. Furthermore, we concrete a ground-truth adversarial example on the maximal safe radius. The detailed algorithms

is in Appendix and the experiment results are demonstrated in Figure 2. Figure 2(a) demonstrates the process of searching for ground-truth adversarial examples in Figure 2(c). The original input is recognized as name"JULES", in which the 12th output frame corresponds to letter "L". We choose eight pixels for perturbation, calculating the upper and lower bounds of the output frame for label "L" and target label "I". The output confidence intervals are separated from each other when perturbation radius $\theta < 0.464$, which indicates the safety of the network. When the perturbation radius increases, the confidence intervals overlap and the network is unsafe. On the radius where the lower bound of the original label meets the upper bound of the target label, we find the ground-truth adversarial in Figure2(c). In this experiment, RNNgo demonstrates its capability for safety analysis of complex practical neural networks. The method can be applied for any recurrent networks, regardless of the detail structure and loss function, as long as the network is Lipschitz continous.

5 Conclusion

We propose a method for safety analysis of RNN, which can provide reachability with guarantee. Furthermore it calculates the maximal safe radius and concretes the ground-truth adversarial example. We demonstrate that it can be deployed in any RNN regardless of complex structure or loss function, as long as the network is Lipschitz constant. Future work includes using parallel computation and adopting accurate Lipschitz estimation scheme to improve the efficiency.

References

Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. *arXiv preprint arXiv:1605.07262*, 2016.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017.

Nicholas Carlini and David Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 1–7. IEEE, 2018.

Souradeep Dutta, Susmit Jha, Sriram Sanakaranarayanan, and Ashish Tiwari. Output range analysis for deep neural networks. *arXiv preprint arXiv:1709.09130*, 2017.

Samuel G Finlayson, Isaac S Kohane, and Andrew L Beam. Adversarial attacks against medical deep learning systems. *arXiv preprint arXiv:1804.05296*, 2018.

Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2018.

Victor Gergel, Vladimir Grishagin, and Alexander Gergel. Adaptive nested optimization scheme for multidimensional global search. *Journal of Global Optimization*, 66(1):35–51, 2016.

Yuan Gong and Christian Poellabauer. Crafting adversarial examples for speech paralinguistics applications. *arXiv preprint arXiv:1711.03280*, 2017.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649. Ieee, 2013.

Vladimir Grishagin, Ruslan Israfilov, and Yaroslav Sergeyev. Convergence conditions and numerical comparison of global optimization methods based on dimensionality reduction schemes. *Applied Mathematics and Computation*, 318:270–280, 2018.

Yanming Guo, Yu Liu, Theodoros Georgiou, and Michael S Lew. A review of semantic segmentation using deep neural networks. *International journal of multimedia information retrieval*, 7(2):87–93, 2018.

Simon Hecker, Dengxin Dai, and Luc Van Gool. Learning driving models with a surround-view camera system and a route planner. *15th European Conference on Computer Vision (ECCV)*, abs/1803.10158, 2018.

Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37:100270, 2020.

Yuval Jacoby, Clark Barrett, and Guy Katz. Verifying recurrent neural networks using invariant inference. *arXiv preprint arXiv:2004.02462*, 2020.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

Ching-Yun Ko, Zhaoyang Lyu, Tsui-Wei Weng, Luca Daniel, Ngai Wong, and Dahua Lin. Popqorn: Quantifying robustness of recurrent neural networks. *arXiv preprint arXiv:1905.07387*, 2019.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Xiaolei Liu, Xiaosong Zhang, Kun Wan, Qingxin Zhu, and Yufei Ding. Towards weighted-sampling audio adversarial example attack. *arXiv preprint arXiv:1901.10300*, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pp. 49–54. IEEE, 2016.

Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. Reachability analysis of deep neural networks with provable guarantees. *arXiv preprint arXiv:1805.02242*, 2018.

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin T Vechev. Fast and effective robustness certification. *NeurIPS*, 1(4):6, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Dmitri Vengertsev and Elena Sherman. Recurrent neural network properties and their verification with monte carlo techniques. 2020.

Weiming Xiang, Hoang-Dung Tran, and Taylor T Johnson. Reachable set computation and safety verification for neural networks with relu activations. *arXiv preprint arXiv:1712.08163*, 2017.

Hongce Zhang, Maxwell Shinn, Aarti Gupta, Arie Gurfinkel, Nham Le, and Nina Narodytska. Verification of recurrent neural networks for cognitive tasks via reachability analysis. *The 24th European Conference on Artificial Intelligence*, 2020.

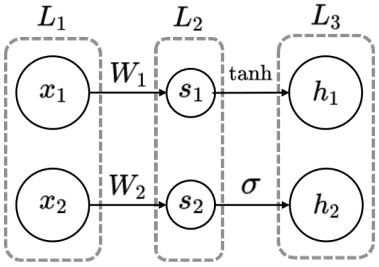

Figure 3: Feed forward layer with different activation functions

A Appendix

A.1 Lipschitz continuity of Layer with different activation functions

Theorem 1. *For a layer that contains n types of activation functions $\sigma_1, \sigma_2, ..., \sigma_n$,assuming that their corresponding inputs are $x_1, x_2, ..., x_n$ and the matrix norms here are submultiplicative, if $\forall i = 1, ..., n$, $\left\| \frac{\partial \sigma_i(x_i)}{\partial x_i} \right\|$ is bounded by a constant θ_i, then $\left\| \frac{\partial w(x)}{\partial x} \right\|$ is also bounded by a constant where $w = \sigma_1, \sigma_2, ..., \sigma_n$ and $x = x_1, x_2, ..., x_n$.*

Proof of 1. By the definition of partial derivative and the fact that activation function σ_i is only connected with x_i where $i = 1, 2, ..., n$, we have

$$\frac{\partial w}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \sigma_1}{\partial x_1} & & \cdots & & 0 \\ & \ddots & & & \\ \vdots & & \frac{\partial \sigma_i}{\partial x_i} & & \vdots \\ & & & \ddots & \\ 0 & & \cdots & & \frac{\partial \sigma_n}{\partial x_n} \end{bmatrix} \tag{1}$$

where $I_{1,i}$ and $I_{2,i}$ are the identity matrices with appropriate sizes. Based on the submultiplicative property of matrix norms, we further have

$$\left\| \frac{\partial w}{\partial \mathbf{x}} \right\| \leq \sum_{i=1}^{n} \|(0 \cdots I_{1,i} \cdots 0)\| \left\| \left(\frac{\partial \sigma_i}{\partial x_i} \right) \right\| \left\| \begin{pmatrix} 0 \\ \vdots \\ I_{2,i} \\ \vdots \\ 0 \end{pmatrix} \right\| \leq \sum_{i=1}^{n} c_{1,i} \theta_i c_{2,i} \tag{2}$$

where $c_{1,i}$ and $c2, i$ are the norm values for the corresponding identify matrices, which are fixed and independent of x_i. Finally, Eqn. (2) concludes that $\left\| \frac{\partial w}{\partial \mathbf{x}} \right\|$ is bounded by a constant. And thus, layers of different activation functions are Lipschitz continuous.

A.2 Safety Evaluation with RNNgo

We present our approach RNNgo for safety analysis of recurrent neural network. We treats the verification as an optimization problem, so that it not only calculates the bounds u and l of the output, but also returns the input x_u and x_l corresponding to the bound value.

A.2.1 Safety Verification of RNN

In the verification process, we perturb only part of the input, and calculate the output bounds through optimization with Lipschitz constant. We describe an original input as $X_0 = \underline{X_0} \cup \overline{X_0}$, in which $\underline{X_0}$ and $\overline{X_0}$ are the unchangeable part and perturbable part respectively. We define the input X

for evaluation as $X = \underline{X_0} \cup \overline{X'}$,with x' representing each element in changeable part X', through lipschitz optimization we calculate

$$l = \min f(\underline{X_0} \cup X'), \text{ s.t. } \quad a \leq x' \leq b. \tag{3}$$

In the Lipschitz optimization, we use zigzag lines to approach the bound l, details of the optimization can be found in Ruan et al. (2018). Through the optimization we can get bounds l and u with error ϵ. In RNNgo, the optimization is performed on confidence values of each label, obtaining the bounds of each label $(l_1, u_1), ..., (l_s, u_s)$. For an given input with label j, RNNgo compares lower bound of $c_j(x)$ with other upper bounds. If $l_j \geq u_k$ for any $k \neq j$, we can conclude that the neural network f is safe with a perturbation $a \leq x' \leq b$

A.2.2 Maximal Safe Radius

When the network f is not safe with certain perturbation, it means we can find a label k with $u_k \geq l_j$. In such case, we reduce the perturbation, i.e. norm ball radius, step by step with binary search until we find the maximal safe norm ball. We add an limitation of perturbation of θ in the optimization 3 and

$$\begin{aligned} & l = \min f(\underline{X_0} \cup X') \\ subject\ to \quad & max(a, x - \theta) \leq x' \leq min(b, x + \theta) \end{aligned} \tag{4}$$

Algorithm 1 Calculation of maximal safe radius for target attack

Input: f, x_0, θ_0, ϵ, original label j, target label k, a, b
Output: maximal safe radius r

$\theta \leftarrow \theta_0, \theta_{min} \leftarrow 0, \theta_{max} \leftarrow (b - a)$
for i in range N **do**
  $l_j, x_j \leftarrow RNNgomin(f, x_0, x', \theta, a, b, j)$
  $u_k, x_k \leftarrow RNNgomax(f, x_0, x', \theta, a, b, k)$
  $i++$
  if $l_j - u_k > \epsilon$ **then**
    $\theta \leftarrow (\theta + \theta_{max})/2$
  else if $u_k - l_j > \epsilon$ **then**
    $\theta \leftarrow (\theta + \theta_{min})/2$
  else
    break
  end if
end for
return x_k, θ

Because of the fixed input length, we choose Lipschitz constant $K = 3$ in the experiment of safety evaluation.

A.3 Experiments

Figure 4 shows the maximal safe radius of a data in MNIST data set.

Table 3 further demonstrate the comparison results of RNNgo and POPQORN.

Figure 5 shows the ablation study restuls of Lipschitz constant K and perturbed pixel number.

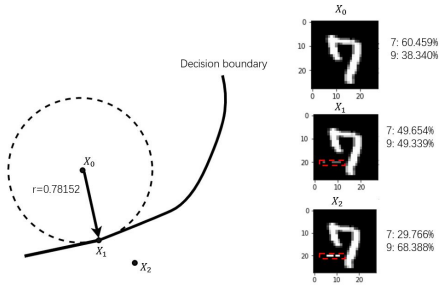

Figure 4: Maximal safe radius and ground-truth adversarial example of an MNNIST data point in a LSTM 7_64 network. X_0 is the original input near the decision boundary, which is classified as "7" with 60.459% confidence. We perturb the eight pixels in the red dotted line box and get the maximal safe radius $r = 0.78$. X_1 is ground-truth adversarial example on the safe radius. X_2 indicates the lower bound on the whole input rage.

Table 3: Averaged safe radius and time cost of RNNgo and POPQORN on verification of networks with different hidden neurons.

Models	DeepAgn		POPQORN	
	safe radius	time	safe radius	time
rnn 7_16	0.5580	117.82s	0.2038	2.14s
rnn 7_32	0.2371	175.92s	0.1340	2.44s
rnn 7_128	0.6633	240.59s	0.1052	4.25s
rnn 7_256	0.6656	187.83s	0.2038	1.89s
lstm 7_16	0.3789	175.11s	0.0007	243.60s
lstm 7_32	0.3461	189.51s	0.0015	256.77s
lstm 7_128	0.3625	256.50s	0.0050	375.85s

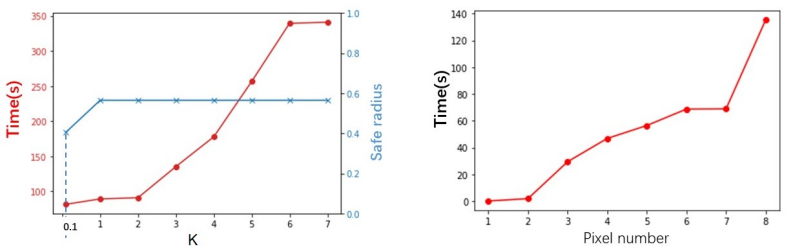

Figure 5: Time cost and safe radius with different Lipschitz constant K and perturbed pixel numbers