POPQORN: Quantifying Robustness of Recurrent Neural Networks

Ching-Yun Ko 1 2  Zhaoyang Lyu 2  Tsui-Wei Weng 3  Luca Daniel 3  Ngai Wong 1  Dahua Lin 2

Abstract

The vulnerability to adversarial attacks has been a critical issue for deep neural networks. Addressing this issue requires a reliable way to evaluate the robustness of a network. Recently, several methods have been developed to compute robustness quantification for neural networks, namely, certified lower bounds of the minimum adversarial perturbation. Such methods, however, were devised for feed-forward networks, e.g. multi-layer perceptron or convolutional networks. It remains an open problem to quantify robustness for recurrent networks, especially LSTM and GRU. For such networks, there exist additional challenges in computing the robustness quantification, such as handling the inputs at multiple steps and the interaction between gates and states. In this work, we propose POPQORN (Propagated-output Quantified Robustness for RNNs), a general algorithm to quantify robustness of RNNs, including vanilla RNNs, LSTMs, and GRUs. We demonstrate its effectiveness on different network architectures and show that the robustness quantification on individual steps can lead to new insights.

1. Introduction

Deep learning has led to remarkable performance gains on a number of tasks, i.e. image classification, speech recognition, and language processing. Nevertheless, recent literature has demonstrated that adversarial examples broadly exist for deep neural networks in these applications (Szegedy et al., 2014; Kurakin et al., 2017; Jia & Liang, 2017; Carlini & Wagner, 2018). A small perturbation that humans are mostly immune to can be crafted to mislead a neural network’s predictions. As deep neural networks have been widely employed in many safety-critical applications (Sharif et al., 2016; Kurakin et al., 2017; Eykholt et al., 2018), it is crucial to know when such models will fail and what can be done to make them more robust to the adversarial attacks.

Studies on the robustness of neural networks mainly fall in two categories: (1) Attack-based approaches – researchers try to design strong adversarial attack algorithms to attack deep neural networks and the robustness is measured by the distortion between successful adversarial examples and the original ones. (2) Verification-based approaches (Katz et al., 2017; Cheng et al., 2017), which aim to find the minimum distortion of neural networks or its lower bounds that are agnostic to attack methods. Existing verification-based methods were mainly devised for feed-forward networks, such as multi-layer perceptron. Robustness verification for recurrent neural networks (RNNs), which have been widely used in speech recognition and natural language processing, has not been systematically investigated.

To verify the robustness of RNNs, we face several new challenges: (1) Some popular RNN formulations, including LSTM and GRU, involve hidden states that are indirectly determined by three to four gates. These nonlinear gates are tightly coupled together, which we refer to as cross-linearity. This substantially complicates the verification of robustness. (2) RNNs are widely adopted for applications with sequential inputs, e.g. sentences or time series. However, previous verification methods typically assume that the inputs were fed into the network at the bottom layer. Hence, they are not directly applicable here. (3) For applications with sequential inputs, imperceptible adversarial examples may correspond to texts with least number of changed words (Gao et al., 2018). Thus it is critical to evaluate the hardness of manipulating one single word (one input frame) instead of all words.

In this paper, we tackle the aforementioned problems by proposing an effective robustness quantification framework called POPQORN (Propagated-output Quantified Robustness for RNNs) for RNNs. We bound the non-linear activation function using linear functions. Starting from the output layer, linear bounds are propagated back to...
the first layer recursively. Compared to existing methods, POPQORN has three important advantages: (1) **Novel** - it is a general framework, which is, to the best of our knowledge, the first work to provide a quantified robustness evaluation for RNNs with robustness guarantees. (2) **Effective** - it can handle complicated RNN structures besides vanilla RNNs, including LSTMs and GRUs that contain challenging coupled nonlinearities. (3) **Versatile** - it can be widely applied in applications including but not limited to computer vision, natural language processing, and speech recognition.

### 2. Background and Related Works

**Adversarial attacks in RNNs.** Crafting adversarial examples of RNNs in natural language processing and speech recognition has started to draw public attentions in addition to the adversarial examples of feed-forward networks in image classifications. Adversarial attacks of RNNs on text classification task (Papernot et al., 2016a), reading comprehension systems (Jia & Liang, 2017), seq2seq models (Cheng et al., 2018) have been proposed in natural language processing application; meanwhile recently (Gong & Poellabauer, 2017) and (Cissé et al., 2017; Carlini & Wagner, 2018) have also demonstrated successful attacks in speech recognition and audio systems. We summarize the RNN attacks in Table 1. (Papernot et al., 2016b; Gong & Poellabauer, 2017; Ebrahimi et al., 2018a;b; Cheng et al., 2018; Carlini & Wagner, 2018) perform gradient-based attacks, while generative adversarial network is used in (Zhao et al., 2018) to generate adversaries. We adapt (Carlini & Wagner, 2017) into C&W-Ada for finding RNN attacks.

**Robustness verification for neural networks.** To safeguard against misclassification under a threat model, verification-based methods evaluate the strengths such that any possible attacks weaker than the proposed strengths will fail. A commonly-used threat model is the norm-ball bounded attacks, wherein strengths of adversaries are quantified by their $l_p$ distance from the original example. Under the norm-ball bounded threat model, determining the minimum adversarial distortion of ReLU networks has been shown to be NP-hard (Katz et al., 2017). Despite the hardness of the problem, fortunately, it is possible to estimate the minimum adversarial distortion (Weng et al., 2018b) or to compute a non-trivial certified lower bound (Hein & Andriushchenko, 2017; Raghunathan et al., 2018; Weng et al., 2018a; Dvijotham et al., 2018; Zhang et al., 2018; Singh et al., 2018; Boopathy et al., 2019). In (Weng et al., 2018b), the authors proposed a robustness score, CLEVER, for neural network image classifiers based on estimating local Lipschitz constant of the networks. We show that it is possible to directly adapt their framework and compute a CLEVER score for RNNs, which is referred to CLEVER-RNN in Section 4.

However, CLEVER score does not come with guarantees and therefore an alternative approach is to compute a non-trivial lower bound of the minimum adversarial distortion. (Hein & Andriushchenko, 2017) and (Raghunathan et al., 2018) proposed to analytically compute the lower bounds for shallow networks (with no more than 2 hidden layers) but their bounds get loose easily when applied to general networks. (Weng et al., 2018a) proposed two algorithms Fast-Lin and Fast-Lip to efficiently quantify robustness for ReLU networks and the Fast-Lin algorithm is further extended to CROWN (Zhang et al., 2018) to quantify for MLP networks with general activation functions. On the other hand, (Dvijotham et al., 2018) proposed a dual-approach to verify NNs with a general class of activation functions. However, their approach compromises the lower bound performance and
the computational efficiency of the “anytime“ algorithm. Recently, (Boopathy et al., 2019) proposes a framework that is capable of quantifying robustness on CNNs with various architectures including pooling layers, residual blocks in contrast to (Singh et al., 2018) that is limited to convolutional architectures without pooling layers.

To the best of our knowledge, there is no prior work to provide robustness verification for RNNs that tackles the aforementioned challenges: the complex feed-back architectures, the sequential inputs, and the cross-linearity of the hidden states. In line with the robustness verification for NNs, this paper is the first to study the robustness properties of RNNs (see Tables 1 and 2 for detailed comparisons). Since it is not straightforward to define a mistake in tasks other than classifications (cf. a mistake in classification tasks refers to a misclassification), we limit the scope of this paper to RNN-based classifiers.

3. POPQORN: Quantifying Robustness of Recurrent Neural Networks

**Overview.** In this section, we show that the output of an RNN can be bounded by two linear functions when the input sequence of the network is perturbed within an $\ell_p$ ball with a radius $\epsilon$. By applying the linear bounds on the non-linear activation functions (e.g. sigmoid and tanh) and the non-linear operations (e.g. cell states in LSTM), we can decouple the non-linear activations and the cross-linearity in the hidden states layer by layer and eventually bound the network output by two linear functions in terms of input$^1$. Subsequently, we show how this theoretical result is used in the POPQORN to compute robustness quantification of an RNN. For ease of illustrations, we start with two motivating examples on a 2-layer vanilla RNN and a 1-layer LSTM network. The complete theorems for general $m$-layer vanilla RNNs, LSTMs, and GRUs are provided in Section A in the appendix.

**Notations.** Let $X_0 = [x^{(1)}_0, \ldots, x^{(m)}_0]$ be the original input data sequence, and let $X = [x^{(1)}, \ldots, x^{(m)}]$ be the perturbed frames of $X_0$ within an $\epsilon$-bounded $\ell_p$-ball, i.e., $x^{(k)} \in \mathbb{B}_p(x^{(k)}_0, \epsilon)$, where superscript “$(k)$” denotes the $k$-th time step. An RNN function is denoted as $F$ and the $j$-th output element as $F_j(X)$. The upper and lower bounds of $F_j(X)$ are denoted as $F^U_j(X)$ and $F^L_j(X)$, respectively. The full notations are summarized in Table 3.

3.1. A 2-layer Vanilla RNN

**Definition.** A many-to-one 2-layer ($m = 2$) RNN reads

\[
F(X) = W^{Fa} a^{(2)} + b^F,
\]

where $F$ is the output, $a^{(k)}$ is the $k$-th hidden state, $\sigma(\cdot)$ is the coordinate-wise activation function, and $W^{aa}, W^{Fa}, b^F, a^0$ are associated model parameters, and shown in Figure 1.

**Ideas.** Based on the above equation, for a fixed $j \in [t]$, we aim at deriving two explicit functions $F^L_j$ and $F^U_j$ such that $\forall X \in \mathbb{R}^{n \times 2}$, $x^{(k)} \in \mathbb{B}_p(x^{(k)}_0, \epsilon)$, the inequality $F^L_j(X) \leq F_j(X) \leq F^U_j(X)$ holds true. To start with, we assume that we know the bounds for pre-activation $y^{(k)} = W^{aa} a^{(k-1)} + W^{ax} x^{(k)} + b^a$ (superscript “$(k)$” denotes the $k$-th layer): $l^{(k)} \leq y^{(k)} \leq u^{(k)}$. We refer the $l$ and $u$ as pre-activation bounds in the remainder of this paper. We show that it is possible to bound every non-linear activation function $\sigma(\cdot)$ in the hidden states by two bounding lines in (1a) and (1b) associated to $l^{(k)}$ and $u^{(k)}$, and that we can apply this procedure recursively from the output $F_j(X)$ to the input $X$ to obtain $F^U_j$ and $F^L_j$.

**Bounding lines.** Two univariate bounding linear functions are defined as follows:

\[
\begin{align*}
\hat{h}^{(k)}_{U,r}(v) &= \alpha^{(k)}_{U,r}(v + \beta^{(k)}_{U,r}), \\
\hat{h}^{(k)}_{L,r}(v) &= \alpha^{(k)}_{L,r}(v + \beta^{(k)}_{L,r}),
\end{align*}
\]

such that $(1a) \leq \sigma(v) \leq (1b)$,

\[
\text{eq.(1b)} \leq \sigma(v) \leq \text{eq.(1a)},
\]

where the $r$ in the subscript implies the dependency of the derived lines on neurons. Both the slopes and intercepts are functions of pre-activation bounds.

**Derivation.** We exemplify how a 2-layer vanilla RNN can be bounded:

\[
F_j(X) = W^{Fa}_{j,1} a^{(2)} + b^F_{j} = W^{Fa}_{j,1} \sigma(y^{(2)}) + b^F_{j}. \tag{2}
\]

We use $s$ upper-bounding lines $h^{(2)}_{j, r}(y^{(2)}_r), r \in [s]$, and also use variables $\lambda^{(2)}_{j, r}$ and $\Delta^{(2)}_{j, r}$ to denote the slopes in front of

![Graphical depiction of a 2-layer many-to-one RNN.](image)

Figure 1. Graphical depiction of a 2-layer many-to-one RNN.
Table 3. Table of Notation

| Notation | Definition | Notation | Definition |
|----------|------------|----------|------------|
| \( l \)  | number of output classes | \( F : \mathbb{R}^{n \times m} \rightarrow \mathbb{R}^l \) | network classifier |
| \( n \)  | size of input frames | \( \mathbf{x}^{(k)} \in \mathbb{R}^n \) | \( \{ K \} \) |
| \( s \)  | number of neurons in a layer | \( F^s(X) : \mathbb{R}^{n \times m} \rightarrow \mathbb{R} \) | \( \mathbb{B}_p(\mathbf{x}_0^{(k)}, \epsilon) \) |
| \( m \)  | number of network layers | \( F^m(X) : \mathbb{R}^{n \times m} \rightarrow \mathbb{R} \) | \( \{ x | \| x - x_0^{(k)} \|_p \leq \epsilon \} \) |
| \( a_0 \) | initial hidden state | \( \gamma_{j}^{(2)} \) | global lower bound of \( F_j(X) \) |
| \( c_0 \) | initial cell state | \( \gamma_{j}^{(2)} \) | global upper bound of \( F_j(X) \) |
| \( x \)  | original input | \( \mathbf{x}_0^{(k)} \) | \( \mathbb{B}_p(\mathbf{x}_0^{(k)}, \epsilon) \) |

\( \lambda_{j,r}^{(2)} \) and intercepts in the parentheses:

\[
\lambda_{j,r}^{(2)} = \begin{cases} 
\alpha_{L,r}^{(2)} & \text{if } W_{F,j}^{a} \geq 0; \\
\alpha_{U,r}^{(2)} & \text{if } W_{F,j}^{a} < 0; 
\end{cases}
\]

and obtain

\[
F_j(X) \leq \left( W_{F,j}^{a} \odot \lambda_{j,r}^{(2)} \right)(y_{r}^{(2)} + \Delta_{r,j}^{(2)}) + b_{j}^{(2)},
\]

\( \odot \) is the Hadamard (i.e. element-wise) product. To simplify notation, we let \( \Lambda_{j,r}^{(2)} := W_{F,j}^{a} \odot \lambda_{j,r}^{(2)} \) and have

\[
F_j(X) \leq \tilde{W}_{F,j}^{a}(a)^{(1)} + \tilde{W}_{F,j}^{ax}(x)^{(2)} + \tilde{b}_{j}^{(2)},
\]

where

\[
\tilde{W}_{F,j}^{a}(a) = \Lambda_{j,r}^{(2)} W_{F,j}^{a}, \quad \tilde{W}_{F,j}^{ax}(x) = \Lambda_{j,r}^{(2)} W_{F,j}^{ax}, \\
\tilde{b}_{j}^{(2)} = \Lambda_{j,r}^{(2)} b + \Delta_{r,j}^{(2)} + b_{j}^{F}.
\]

Substituting \( a^{(1)} \) with its definition yields

\[
F_j(X) \leq \tilde{W}_{F,j}^{a}(a) + \tilde{W}_{F,j}^{ax}(x) + \tilde{b}_{j}^{(2)}. \tag{3}
\]

Note that Equations (2) and (3) are in similar forms. Thus we can bound \( \sigma(y_{r}^{(2)}) \) by \( s \) linear functions \( \lambda_{j,r}^{(2)}(y_{r}^{(2)}), r \in [s] \) and use \( \lambda_{j,r}^{(1)} \) and \( \Delta_{r,j}^{(1)} \) to denote slopes in front of \( y_{r}^{(1)} \) and intercepts in the parentheses:

\[
\lambda_{j,r}^{(1)} = \begin{cases} 
\alpha_{L,r}^{(1)} & \text{if } W_{F,j}^{a} \geq 0; \\
\alpha_{U,r}^{(1)} & \text{if } W_{F,j}^{a} < 0; 
\end{cases}
\]

\( \Delta_{r,j}^{(1)} = \begin{cases} 
\beta_{L,r}^{(1)} & \text{if } W_{F,j}^{a} \geq 0; \\
\beta_{U,r}^{(1)} & \text{if } W_{F,j}^{a} < 0; 
\end{cases}\]

and let \( \Lambda_{j,r}^{(1)} := \tilde{W}_{F,j}^{a}(a)^{(1)} \). Then we have

\[
F_j(X) \leq \Lambda_{j,r}^{(1)} W_{F,j}^{a}(a)^{(0)} + \sum_{z=1}^{2} \Lambda_{j,r}^{(2)} W_{F,j}^{ax}(x)^{(z)} \\
+ \sum_{z=1}^{2} \Lambda_{j,r}^{(2)} (b + \Delta_{r,j}^{(2)} + b_{j}^{F}).
\]

So far we have derived an explicit linear function on the right-hand side of the above inequality. We denote it herein as \( F^L_j(X) \) and have \( F_j(X) \leq F^L_j(X), \forall X \in \mathbb{R}^{n \times 2} \) where \( x^{(k)} \in \mathbb{B}_p(\mathbf{x}_0^{(k)}, \epsilon) \). A closed-form global upper bound \( \gamma_{j}^{(2)} \) can be obtained naturally by maximizing \( F^U_j(X) \) for all possible \( X \), which can be directly solved by applying Holder’s inequality. Hence we obtain

\[
\gamma_{j}^{U} = \Lambda_{j,r}^{(1)} W_{F,j}^{a}(a)^{(0)} + \sum_{z=1}^{2} \epsilon \| \Lambda_{j,r}^{(2)} W_{F,j}^{ax} \|_q \\
+ \sum_{z=1}^{2} \Lambda_{j,r}^{(2)} (b + \Delta_{r,j}^{(2)} + b_{j}^{F}).
\]

An explicit function \( F^L_j \) and a closed-form global lower bound \( \gamma_{j}^{L} \) can also be found through similar steps and by minimizing \( F^L_j(X) \) instead. During the derivation, we have limited perturbations to be uniform across input frames, which assemble noises in cameras. However, the above method is also applicable to non-uniform distortions. For example, we can also certify bounds for distortions on parts of the input frames, which on the other hand are of more interests in natural language processing tasks.

3.2. A 1-layer Long Short-term Memory Network

For an RNN with LSTM units, we can also derive analytic upper-bounding and lower-bounding functions. In this example, we inherit the notations in Table 3, and exemplify through a 1-layer \((m = 1)\) LSTM as shown in Figure 2.

LSTM. The following updating equations are adopted:

- Input gate: \( i^{(k)} = \sigma(W_{ix}^{i} x^{(k)} + W_{ir}^{i} a^{(k-1)} + b^{i}) \);
- Forget gate: \( f^{(k)} = \sigma(W_{fx}^{f} x^{(k)} + W_{fr}^{f} a^{(k-1)} + b^{f}) \);
- Cell gate: \( g^{(k)} = \tanh(W_{gx}^{g} x^{(k)} + W_{gr}^{g} a^{(k-1)} + b^{g}) \);
- Output gate: \( o^{(k)} = \sigma(W_{ox}^{o} x^{(k)} + W_{or}^{o} a^{(k-1)} + b^{o}) \);
- Cell state: \( c^{(k)} = f^{(k)} \odot c^{(k-1)} + i^{(k)} \odot g^{(k)} \);
- Hidden state: \( a^{(k)} = o^{(k)} \odot \tanh(c^{(k)}) \).

Again, the \( \sigma(\cdot) \) denotes the coordinate-wise sigmoid function and \( \tanh(\cdot) \) is the coordinate-wise hyperbolic tangent function. Output \( F \) of the network is determined by

\[
F(X) = W_{Fa}^{a} a^{(m)} + b^{F},
\]

and we have pre-activations as

\[
\begin{align*}
\gamma_{i}^{(k)} &= W_{ix}^{i} x^{(k)} + W_{ir}^{i} a^{(k-1)} + b^{i}; \\
\gamma_{f}^{(k)} &= W_{fx}^{f} x^{(k)} + W_{fr}^{f} a^{(k-1)} + b^{f}; \\
\gamma_{o}^{(k)} &= W_{ox}^{o} x^{(k)} + W_{or}^{o} a^{(k-1)} + b^{o}.
\end{align*}
\]
Ideas. When deriving the lower bounds and upper bounds of LSTMs, we have also made an important assumption: we assume we know the bounds for hidden/cell states. This assumption, as well as the assumption made when deriving for vanilla RNNs and GRUs, can be easily fulfilled as the bounds of hidden/cell states are available by similar derivations as will be shown below. We refer readers to Theorem A.2 (vanilla RNNs), Corollary A.3/ Theorem A.4 (LSTMs) and Corollary A.6 (GRUs) in the appendix for details.

Recalling that in Section 3.1, we aim at bounding the non-linearity using univariate linear functions. Final bounds are obtained by recursively propagating the linear bounds from the output layer back to the first layer. Here when analyzing the bounds for an LSTM, we adopt a similar approach of propagating linear bounds from the last layer to the first. However, different from the vanilla RNN, the difficulty of reaching this goal lies in bounding more complex nonlinearities. In an LSTM, we cope with two different non-linear functions:

\[ \sigma(v)z \quad \text{and} \quad \sigma(v) \tanh(z), \]

both of which are dependent on two variables and are cross terms of varied gates. To deal with this, we extend our previous ideas to using \textit{planes} instead of \textit{lines} to bound these cross terms. The graphical illustration of the bounding planes is shown in Figure 3.

Bounding planes. With bounds of hidden/cell states, we can gather bounds: \( \|v^{\text{gate}(1)}\| \leq \|y^{\text{gate}(1)}\| \leq \|u^{\text{gate}(1)}\| \) where gate = \{i, f, g, o\} and superscript “(1)” denotes the layer we are at. As both variables are bounded as above and compactness is a continuous invariant, we are guaranteed to have at least two bounding planes for the non-linearities. For example, for \( \sigma(v)z \) we have planes:

\[ h_{\text{cross}(1)}^U,v,r = \max(\sigma(v)z), \quad h_{\text{cross}(1)}^L,v,r = \min(\sigma(v)z). \]

In practice, we use the following planes:

\[ h_{\text{cross}(1)}^U,v,r (v, z) = \alpha_{U,v,r}^{\text{cross}(1)} v + \beta_{U,v,r}^{\text{cross}(1)} z + \gamma_{U,v,r}^{\text{cross}(1)}, \quad (4a) \]

\[ h_{\text{cross}(1)}^L,v,r (v, z) = \alpha_{L,v,r}^{\text{cross}(1)} v + \beta_{L,v,r}^{\text{cross}(1)} z + \gamma_{L,v,r}^{\text{cross}(1)}, \quad (4b) \]

To bound Equation (5), we use \( s \) upper-bounding planes \( h_{U,v,r}^{\text{oc}(1)}(y^{(1)},c^{(1)}), r \in [s], \) and also define \( \lambda_{oc}^{(1)}(j,r), \Delta_{oc}^{(1)}(j,r) \) and \( \varphi_{oc}^{(1)}(j,r) \) in the parentheses:

\[ F_j(X) = W_j^{Fa} a^{(1)} + b_j^F, \]

\[ = W_j^{Fa} [\sigma(y^{(1)}) \circ \tanh(c^{(1)})] + b_j^F. \quad (5) \]

We thus formulate the task of finding bounding planes as a constrained optimization problem and use gradient descent method to solve it (see Section A.4 in the appendix for details).

Derivation. We now exemplify how a 1-layer LSTM can be bounded:

\[ F_j(X) = W_j^{Fa} a^{(1)} + b_j^F, \]

\[ = W_j^{Fa} [\sigma(y^{(1)}) \circ \tanh(c^{(1)})] + b_j^F. \quad (5) \]

and obtain

\[ F_j(X) \leq \left( W_j^{Fa} \circ \lambda_{oc}^{(1)}(j,r) \right) y^{(1)} + \left( W_j^{Fa} \circ \Delta_{oc}^{(1)}(j,r) \right) c^{(1)} \]

\[ + \sum_{r=1}^s \left( W_j^{Fa} \circ \varphi_{oc}^{(1)}(j,r) \right) + b_j^F. \]
POPQORN: Quantifying Robustness of Recurrent Neural Networks

Table 4. Quantified robustness bounds for various RNNs.

| Networks       | γ_j^L ≤ F_j ≤ γ_j^U | Closed-form formulas |
|----------------|-----------------------|----------------------|
| Vanilla RNN   | Upper bounds γ_j^L    | \( \sum_{k=1}^{m} \epsilon \| W_j^k \| F_x + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_y + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_z \) |
|                | Lower bound γ_j^L     | \( \sum_{k=1}^{m} \epsilon \| W_j^k \| F_x + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_y + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_z \) |
| LSTM          | Upper bounds γ_j^L    | \( \sum_{k=1}^{m} \epsilon \| W_j^k \| F_x + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_y + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_z \) |
|                | Lower bound γ_j^L     | \( \sum_{k=1}^{m} \epsilon \| W_j^k \| F_x + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_y + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_z \) |
| GRU           | Upper bounds γ_j^L    | \( \sum_{k=1}^{m} \epsilon \| W_j^k \| F_x + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_y + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_z \) |
|                | Lower bound γ_j^L     | \( \sum_{k=1}^{m} \epsilon \| W_j^k \| F_x + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_y + \sum_{k=1}^{m} \epsilon \| W_j^k \| F_z \) |

Remark: see Section A in the appendix for detailed proofs and definitions.

For simplicity, we further collect the summing weights of \( y^{oc}(1) \) and \( c^{(1)} \) into row vectors \( \Lambda_{\Delta;j,:}^{oc(1)} \) and \( \Lambda_{\Delta;j,:}^{oc(1)} \), and constants (excepts \( b^F \)) into \( \Lambda_{\Delta;j,:}^{oc(1)} \). Similarly, \( \Lambda_{\Delta;j,:}^{ig} \) can also be found through similar steps such that \( \gamma_j^L \leq F_j^L(X) \leq F_j(X) \), \( \forall X \in \mathbb{R}^{n \times m} \) where \( x^{(k)} \in \mathbb{B}_p(x_0^{(k)}, \epsilon) \). The derivations for a GRU follow similarly, and are put into the appendix (Section A.7) due to space constraint.

3.3. Robustness Quantification Algorithm

As summarized in Table 4, given a trained vanilla RNN, LSTM or GRU, input sequence \( x_0 \in \mathbb{R}^{n \times m} \), \( l_p \) ball parameters \( p \geq 1 \) and \( \epsilon \geq 0 \), for \( \forall j \in [t] \), \( 1/q = 1 - 1/p \), there exist two fixed values \( \gamma_j^L \) and \( \gamma_j^U \) such that \( \forall X \in \mathbb{R}^{n \times m} \) where \( x^{(k)} \in \mathbb{B}_p(x_0^{(k)}, \epsilon) \), the inequality \( \gamma_j^L \leq F_j(X) \leq \gamma_j^U \) holds true. Now suppose the label of the input sequence is \( j \), we aim at utilizing the uniform global bounds in Table 4 to find the largest possible lower bound \( \gamma_j^L \) of untargeted attacks. We formalize the objective and constraints as follows:

\[ \gamma_j^L = \max \epsilon, \text{ s.t. } \gamma_j^L(\epsilon) \geq \gamma_j^U(\epsilon), \forall i \neq j. \]

To verify the largest possible lower bound \( \epsilon \) of targeted attacks (target class be \( i \)), we solve the following:

\[ \tilde{\epsilon}(i, j) = \max \epsilon, \text{ s.t. } \gamma_j^L(\epsilon) \geq \gamma_i^U(\epsilon). \]
One can then verify that $\epsilon_j = \min_{i \neq j} \epsilon(i, j)$. Recalling from the derivations in Sections 3.1 and 3.2, slopes and intercepts utilized are functions of the input ranges. Therefore, the global lower bounds $\gamma^l_j$ and upper bounds $\gamma^u_j$ are also implicitly dependent on $\epsilon$, preventing the above optimization problems from having an analytic solution. In this work, we conduct binary search procedures to compute the largest possible $\epsilon_j$ ($\hat{\epsilon}$). In practice, we start by initializing the $\epsilon$ to be $\epsilon_0 > 0$ and deriving upper and lower bounds according to Table 4. These are followed by a condition check that governs the next step. If $\exists i \neq j$ such that $\gamma^u_i \geq \gamma^l_j$, then we decrease $\epsilon$; otherwise, we increase $\epsilon$. We repeat the above procedure until a predefined maximum iterations are reached or when convergence is achieved. Proposed algorithms are summarized in Section A.2 and A.5 in the appendix.

4. Experiments

Methods. The proposed POPQORN is used to quantify robustness for all models herein. To the best of our knowledge, there is no previous work done on quantifying robustness with guarantees for RNNs. Therefore, we compare our results with CLEVER score (Weng et al., 2018b) (an estimation of the minimum adversarial distortion) and C&W attack (Carlini & Wagner, 2017) (an upper bound of the minimum adversarial distortion). We emphasize on analyzing the characteristics of the certified bounds obtained by POPQORN and new insights they lead to. For comparison, we adapt CLEVER score to accommodate sequential input structure and denote it as CLEVER-RNN. Specifically, let $j$ and $i$ be the true and target labels, respectively. Assume $g(X) = F_i(X) - F_j(X)$ is a Lipschitz function, and define $L_i^\epsilon = \max \{ \| \nabla_{g_i}(X) \| : X(k) \in B_p(X(k), \epsilon_0), \forall k \in [m] \}$ where $\nabla_{g_i}(X) = (\partial g_i(X) / \partial x_i^1, \ldots, \partial g_i(X) / \partial x_i^n)^T$. CLEVER-RNN score is then given as min $\{ L_i^\epsilon(X(k), \epsilon_0) \}^3$. We also adapt C&W attack for our task and denote it as C&W-Ada$^4$. The adapted C&W-Ada puts higher weights on finding a successful attack, and prioritizes the minimization of distortion magnitude when an attack is found. We use the maximum perturbation of all frames as the C&W score. We implement our algorithm using PyTorch to enable the use of GPUs. Using a server consisting of 16 NVIDIA Tesla K80 GPUs, it took about 4 hours to calculate certified bounds for 1000 samples in LSTMs with 4 frames, and one day in LSTMs with 14 frames. Except quantifying for LSTMs, the remaining experiments could be finished in about 10 hours using a single NVIDIA TITAN X GPU. More experimental details are given in the appendix Section B.

Experiment I. In this experiment, we evaluate POPQORN and other baselines on totally 12 vanilla RNNs and LSTMs trained on the MNIST dataset. Table 5 gives the certified lower bounds of models found by POPQORN, together with uncertified CLEVER-RNN scores, and upper bounds found by C&W-Ada attacks. Bounds obtained by POPQORN, CLEVER-RNN, C&W-Ada increase with numbers of hidden neurons $s$ in RNNs and LSTMs, and generally decrease as the number of network layers $m$ grows$^5$. The uncertified bounds computed by CLEVER-RNN are similar to those found by C&W-Ada attacks, yet it should be noted that these bounds are without guarantees. A supplementary comparison among bounding techniques (2D bounding planes, 1D bounding lines, constants) for LSTM evaluations is provided in Appendix Section B.1 (accompanied with theorems in Section A.6).

Experiment II. Next, we evaluate the proposed POPQORN quantification on LSTMs trained on the MNIST sequence dataset$^6$. Specifically, we compute the POPQORN bound on only one single input frame (i.e. we fix all input frames but one and derive certified bound for distortions on that frame). After calculating bounds of all input frames, we identify the frames with minimal bounds and call them sensitive strokes.

$^3$CLEVER score is defined by $\min \{ g(X[k]), \epsilon_0 \}$, where $L_i^\epsilon = \max \{ \| \nabla_{g_i}(X) \| : X(k) \in B_p(X(k), \epsilon_0) \}$. We refer readers to the appendix Sec. B.2 for the details.

$^4$We refer readers to the appendix Sec. B.3 for the details.

$^5$The provable safety region ($L_i^\epsilon$ balls) provided by POPQORN are distributed across framelets. The overall distortion allowed for an input sample is computed by $m^{1/p} \epsilon$.

$^6$Freely available at https://edwin-de-jong.github.io/blog/mnist-sequence-data/
Table 5. (Experiment I) Averaged bounds and standard deviations (-) of POPQORN and other baselines on MNIST dataset. POPQORN is the proposed method, CLEVER-RNN and C&W-Ada are adapted from (Weng et al., 2018b) and (Carlini & Wagner, 2017), respectively.

| Network | $l_\infty$ norm | Certified | Uncertified | Attack | Certified | Uncertified | Attack |
|---------|-----------------|-----------|-------------|--------|-----------|-------------|--------|
| RNN     | $l_\infty$        | 0.0105/0.0047 | 0.0851/0.0398 | 0.2561/0.1372 | 0.0087/0.0018 | 0.1250/0.0594 | 0.3267/0.1801 |
| 4 layers | $l_\infty$        | 0.2052/0.0680 | 0.8860/0.3920 | 1.3768/0.6076 | 0.2056/0.0119 | 0.8684/0.3172 | 0.5278/0.4218 |
| 32 hidden nodes | $l_\infty$        | 1.6151/0.2689 | 5.0033/2.6034 | 5.1502/2.4726 | 0.1759/0.0326 | 2.0618/1.3126 | 2.1149/0.9671 |
| LSTM    | $l_\infty$        | 0.0127/0.0054 | 0.1069/0.0563 | 0.2587/0.1314 | 0.0122/0.0055 | 0.1070/0.0556 | 0.2546/0.1201 |
| 4 layers | $l_\infty$        | 0.2487/0.0592 | 1.2006/0.5240 | 1.7072/0.6757 | 0.2361/0.0636 | 0.8302/0.4714 | 1.3321/0.5529 |
| 64 hidden nodes | $l_\infty$        | 1.3199/0.2984 | 6.4128/3.0360 | 6.8347/3.1543 | 1.1913/0.3035 | 4.0658/2.2761 | 4.9698/2.2749 |

Figure 5. (Experiment II) Heat maps of certified 2-norm bounds computed by POPQORN. Left (a): an example of digit “1”. Right (b): an example of digit “4”. Arrows indicate the order of strokes. records the relative displacement from the previous point. Heat maps are used to track the changes in sensitivities (quantified robustness scores) with strokes. We identify the starting point in Figure 5(a) with a circle. Notably, this point has a relatively big certified bound. The corresponding example is the point with the highest bound. In the second example, the question is classified as “ENTY” (entity). Correspondingly, name is among the three most sensitive words, which is consistent with human cognition. In the second example, the question is classified as “LOC” (location). Again, mountain is shortened in the top three sensitive words.

Figure 6. (Experiment III) Two examples in the question classification task. The upper row gives the sample sentence; the lower row shows the POPQORN (2-norm) lower bounds of individual words. “ENTY” and “LOC” represent entity and location, respectively.

(words), which guarantee robust classifier decisions as long as the perturbation magnitude in the word embedding space is smaller than the certificate. The 3 most sensitive words (words with 3 smallest bounds) are underlined. In the first example, the question is classified as “ENTY” (entity). Correspondingly, name is among the three most sensitive words, which is consistent with human cognition. In the second example, the question is classified as “LOC” (location). Again, mountain is shortened in the top three sensitive words. More examples are provided in the appendix Section B.4. Such observed consistency suggests POPQORN’s potential in distinguishing the importance of different words.

5. Conclusion

This paper has proposed, for the first time, a robustness quantification framework called POPQORN that handles various RNN architectures including vanilla RNNs, LSTMs and GRUs. The certified bound gives a guaranteed lower bound of the minimum distortion in RNN adversaries, in contrast to the upper bound suggested by attacks. Experiments on different RNN tasks have demonstrated that POPQORN can compute non-trivial certified bounds, and provide insights on the importance and sensitivity of a single frame in sequence data.
Acknowledgment

The authors would like to thank Zhuolun Leon He for useful discussion. This work is partially supported by the Big Data Collaboration Research grant from SenseTime Group (CUHK Agreement No. TS1610626), the General Research Fund (Project 14236516, 17246416) of the Hong Kong Research Grants Council, MIT-IBM program, MIT-Skoltech program, and MIT-SenseTime program.

References

Boopathy, A., Weng, T.-W., Chen, P.-Y., Liu, S., and Daniel, L. Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In AAAI, Jan 2019.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In 2017 IEEE Symposium on Security and Privacy, pp. 39–57, 2017.

Carlini, N. and Wagner, D. A. Audio adversarial examples: Targeted attacks on speech-to-text. CoRR, abs/1801.01944, 2018.

Cheng, C.-H., Nührenberg, G., and Ruess, H. Maximum resilience of artificial neural networks. In International Symposium on Automated Technology for Verification and Analysis, pp. 251–268, 2017.

Cheng, M., Yi, J., Zhang, H., Chen, P., and Hsieh, C. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. CoRR, abs/1803.01128, 2018.

Cissé, M., Adi, Y., Neverova, N., and Keshet, J. Houdini: Fooling deep structured prediction models. CoRR, abs/1707.05373, 2017.

Dvijotham, K., Stanforth, R., Gowal, S., Mann, T., and Kohli, P. A dual approach to scalable verification of deep networks. UAI, 2018.

Ebrahimi, J., Lowd, D., and Dou, D. On adversarial examples for character-level neural machine translation. In COLING, pp. 653–663, 2018a.

Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. Hotflip: White-box adversarial examples for text classification. In ACL, pp. 31–36, 2018b.

Eykholt, K.,Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. Robust physical-world attacks on deep learning visual classification. In CVPR, pp. 1625–1634, 2018.

Gao, J., Lanchantin, J., Soffa, M. L., and Qi, Y. Black-box generation of adversarial text sequences to evade deep learning classifiers. In 2018 IEEE Security and Privacy Workshops, pp. 50–56, 2018.

Gong, Y. and Poellabauer, C. Crafting adversarial examples for speech paralinguistics applications. CoRR, abs/1711.03280, 2017.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. In NIPS, 2017.

Jia, R. and Liang, P. Adversarial examples for evaluating reading comprehension systems. In EMNLP, pp. 2021–2031, 2017.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In CAV, pp. 97–117, 2017.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. ICLR Workshop, 2017.

Li, X. and Roth, D. Learning question classifiers. In COLING, pp. 1–7, 2002.

Papernot, N., McDaniel, P., Swami, A., and Harang, R. Crafting adversarial input sequences for recurrent neural networks. In MILCOM, pp. 49–54, 2016a.

Papernot, N., McDaniel, P. D., Swami, A., and Harang, R. E. Crafting adversarial input sequences for recurrent neural networks. MILCOM, pp. 49–54, 2016b.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. ICLR, 2018.

Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 1528–1540, 2016.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. Fast and effective robustness certification. In NIPS, pp. 10825–10836, 2018.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. ICLR, 2014.

Weng, T.-W., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Boning, D., Dhillon, I. S., and Daniel, L. Towards fast computation of certified robustness for relu networks. ICML, 2018a.

Weng, T.-W., Zhang, H., Chen, P.-Y., Yi, J., Su, D., Gao, Y., Hsieh, C.-J., and Daniel, L. Evaluating the robustness of neural networks: An extreme value theory approach. In ICLR, 2018b.
Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. In *NIPS*, pp. 4944–4953. 2018.

Zhao, Z., Dua, D., and Singh, S. Generating natural adversarial examples. In *ICLR*, 2018.