STR-Cert: Robustness Certification for Deep Text Recognition on Deep Learning Pipelines and Vision Transformers

Daqian Shao daqian.shao@cs.ox.ac.uk
University of Oxford
Lukas Fesser lukas_fesser@fas.harvard.edu
Harvard University
Marta Kwiatkowska marta.kwiatkowska@cs.ox.ac.uk
University of Oxford

Abstract

Robustness certification, which aims to formally certify the predictions of neural networks against adversarial inputs, has become an important tool for safety-critical applications. Despite considerable progress, existing certification methods are limited to elementary architectures, such as convolutional networks, recurrent networks and recently Transformers, on benchmark datasets such as MNIST. In this paper, we focus on the robustness certification of scene text recognition (STR), which is a complex and extensively deployed image-based sequence prediction problem. We tackle three types of STR model architectures, including the standard STR pipelines and the Vision Transformer. We propose STR-Cert, the first certification method for STR models, by significantly extending the DeepPoly polyhedral verification framework via deriving novel polyhedral bounds and algorithms for key STR model components. Finally, we certify and compare STR models on six datasets, demonstrating the efficiency and scalability of robustness certification, particularly for the Vision Transformer.

1 Introduction

While deep learning has achieved remarkable performance in a broad range of tasks, such as image classification and natural language processing, deep neural networks (DNNs) are known to be vulnerable to adversarial attacks [11, 60, 55]. These are inputs that, whilst originally predicted correctly, are misclassified after adding slight and often imperceptible perturbations, illustrated in 1. The discovery of these vulnerabilities has motivated a multitude of studies on the robustness of neural networks [7, 58]. One popular

Figure 1: An *adversarial example* from the IC15 dataset, where the predicted text changes under small L_∞ perturbations. The original image is also shown to be not robust by STR-Cert.

direction is neural network *certification* (also called *verification* in the literature), which aims to automatically prove that a network satisfies properties of interest, such as robustness to perturbations [45, 58], safety guarantees [1] and other pre- and post-conditions. Although recent advances have made certification an important tool for analyzing and reasoning about neural networks, its scalability, precision and support for complex deep learning models and tasks still remain key challenges [22] in the field. Most of the existing certification methods focus on simple architectures, such as fully-connected networks (FCNNs) and convolutional networks (CNNs). Some progress has recently been achieved for recurrent networks (RNNs) [36, 21], CNN-RNN models [54] and Transformers [42, 5, 24], but typically only for simple tasks and datasets such as MNIST classification (see the taxonomy of Li *et al.* [22]). Thus, there is a lack of efficient methodologies and experimental evaluation for robustness certification of complex decision pipelines used in real-world applications on large benchmark datasets, often due to the poor scalability of the underlying techniques.

In this work, we focus on scene text recognition (STR), which is the task of recognizing text from natural images. STR systems are extensively deployed in businesses, banks, and law enforcement for applications such as document information extraction and number plate recognition, but are prone to adversarial attacks [23, 57]. Unlike general object recognition, STR is an image-based sequence recognition problem that requires the model to predict a sequence of object labels given an image. To achieve good performance, STR systems are built as sophisticated pipelines, dramatically increasing the complexity and difficulty of the task, precluding direct application of existing methodologies. In this work, we certify three types of STR model architectures including the standard STR architecture pipeline [39, 52] (see 2.2) and the Vision Transformer [2]. Building on and

significantly extending the DeepPoly polyhedral verification [43, 36] framework, we develop STR-Cert, an efficient robustness certification methodology that scales to these complex STR pipelines. Experiments are conducted on six STR datasets to demonstrate the usability of our methods in practice, where we compare and provide insight on the robustness of different STR models.

Novel contributions Our contributions are as follows.

- We propose STR-Cert, an efficient and scalable robustness certification method, and to the best of our knowledge, the first method to certify the robustness of text recognition models and the Vision Transformer.
- We derive novel polyhedral bounds and algorithms to certify key components of the STR models such as the CTC decoder, the Softmax function, patch embedding, and the spatial transformation network.
- We significantly extend the polyhedral verification framework to implement STR-Cert, which can certify 3 types of STR architectures including the Vision Transformer.
- We extensively certify the robustness of the STR models on 6 datasets, providing insights and comparisons between different architectures.

Related Works Since neural networks can be encoded as sets of constraints, robustness to adversarial perturbations can be solved exactly and optimally, using mixed integer programming [45, 5], branch and bound [49], or satisfiability modulo theory [19]. These methods are referred to as complete verification. However, complete verification is NP-complete even for a simple ReLU network [44]. Another group of methods considers a relaxation to the verification problem: incomplete verification, which guarantees to output "not verified" if the input is not safe, but not vice versa. Many incomplete verification methods can be viewed as applying convex relaxations for non-linear activation functions [37], including those based on duality [53, 9], polyhedron abstraction [58, 43, 56], zonotope abstraction [5], layer-by-layer reachability analysis [48, 51], multi-neuron relaxations [30] and semi-definite relaxations [34]. Robustness verification can also be achieved through an analysis of local Lipschitz constants [38, 16, 59]. In this work, we focus on incomplete verification through polyhedral abstraction, which abstracts the input domain and propagates the abstract domain through the neural network via abstract transformations. It is faster than semi-definite and

multi-neuron relaxations, and more precise than other weakly relational domains like zonotopes and interval bounds [22], offering a good trade-off between scalability and precision. CROWN [58] and DeepPoly [43] are the first methods to utilize polyhedral abstraction to certify FCNNs, and were further developed into certification methods for CNNs with general activation functions [6], RNNs [36], and NLP Transformers [42], including the self-attention mechanism and the Softmax function. In addition to architecture support, recent improvements in scalability and precision include incorporating BaB [49, 41], GPU parallelism [29], and optimized polyhedral relaxations [56]. However, most robustness verification methods are evaluated on MNIST and CIFAR-10, recently scaling up to Tiny ImageNet [22], while only a few consider tasks other than image classification. Speech recognition [36, 31] and sentiment analysis [42, 21] are studied in the context of RNN and Transformer certification, while existing certification methods have been applied to object detection/segmentation [10] and reinforcement learning [3] problems. To the best of our knowledge, STR-Cert is the first methodology for certifying STR networks — complex deep learning pipelines with sequence outputs — based on novel algorithms, which we evaluate on six commonly used STR benchmark datasets.

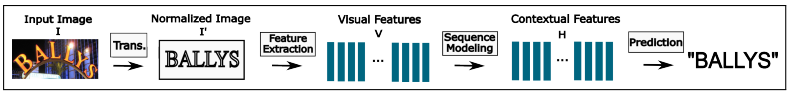
2 Background

Let $\mathcal{N}$ be a neural network for STR tasks with T prediction frames and a set $\mathcal{C}$ of classes that includes all letters and numbers, special symbols, and special tokens such as *blank* and *end-of-sentence* tokens depending on the architecture. Given an image input $I \in \mathbb{R}^{H \times W \times C}$, where H, W and C are height, weight, and channels of the image, respectively, denote the output of the network by $\mathcal{N}(I)$, where for each frame $t \in [1..T]$ and class $c \in \mathcal{C}$, $\mathcal{N}_{t,c}(I)$ is the logits output for predicting class c in frame t. The network output $\mathcal{N}(I)$ is then fed into different decoders $\mathcal{D}$ depending on the architecture to retrieve the sequence of characters $\mathcal{D}(\mathcal{N}(I))$ as the text in the image.

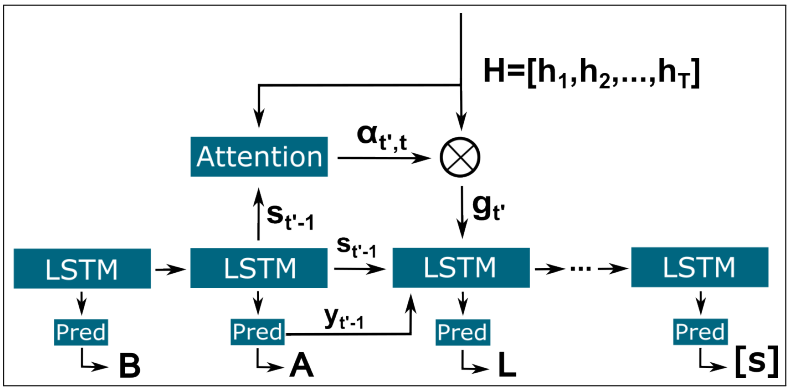
2.1 Robustness Certification for Neural Networks

For an input I and a STR network $\mathcal{N}$ with its decoder $\mathcal{D}$, an adversary can perturb the input to a set $\mathbb{S}_I$ of possible perturbations, denoted as *adversarial region*, before feeding into $\mathcal{N}$. The *adversarial robustness* problem [43] checks whether, for all possible perturbations $I' \in \mathbb{S}_I$, the predicted sequence and that for the original unperturbed input coincide, i.e.,

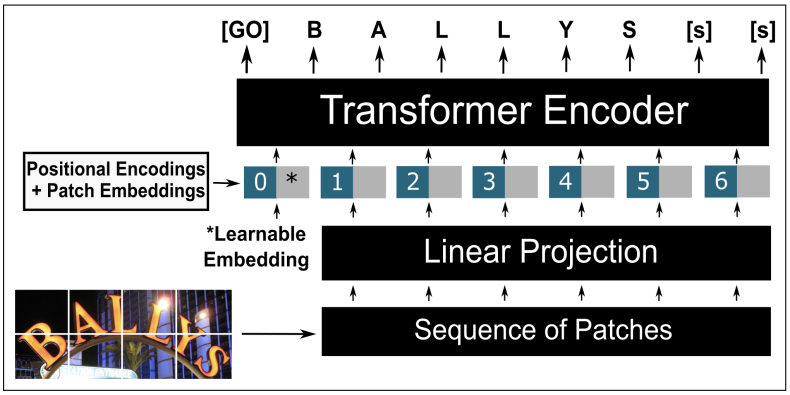
$\mathcal{D}(\mathcal{N}(I')) = \mathcal{D}(\mathcal{N}(I))$. If so, it can be certified that the adversary cannot alter the classification by picking inputs from $\mathbb{S}_I$. In this work, we focus on certifying robustness for adversarial regions that can be represented by the Cartesian product of interval constraints, i.e., $\mathbb{S} = \bigtimes_{i=1}^{H \cdot W \cdot C} [l_i, u_i]$ with $l_i, u_i \in \mathbb{R} \cup \{-\infty, \infty\}$, which allows the certification against the widely used L_∞-norm attacks up to some *perturbation budget*.



(a) The standard four-stage pipeline for STR.



(b) The attention decoder.



(c) Vision Transformer for scene text recognition (ViTSTR).

Figure 2: Three types of common STR model architectures we consider in this work.

2.2 Standard STR Architectures

We consider three STR architectures, of which two are instances of standard STR architectures [39, 52], and the third is the popular Vision Transformer [2]. In this section, we introduce the standard STR architectures, which consist of four stages [4] as shown in 2a:

1. **Transformation** rectifies and normalizes curved and angled text in the image through a variant of the spatial transformer network (STN [15]);
2. **Feature extraction** maps the rectified image into a sequence of T visual features through a CNN network, denoted by $\mathcal{V} = [v_1, v_2, ..., v_T]$;
3. **Sequence modeling** processes the visual features and produces the contextual features $\mathcal{H} = [h_1, h_2, ..., h_T]$ using a long short-term memory (LSTM) network;
4. **Prediction** decodes the contextual features to predict the output text. The decoder can be a connectionist temporal classification (CTC) decoder [12], or an attention decoder [52], both introduced below.

The first three stages constitute the STR network $\mathcal{N}$ with $\mathcal{N}(I) = \mathcal{H}$, whereas the last stage is the decoder $\mathcal{D}$.

2.2.1 Transformation Stage

In STR models, thin plate spline (TPS) transformation [40, 25], a variant of STN [15], is typically applied to rectify the input images. Given an image I, TPS adopts a localization CNN network to produce *fiducial points* Θ that generate the transformation grid. Then, let

$$\mathcal{T} = \left(\Delta_{\Theta'}^{-1} \begin{bmatrix} \Theta^\top \\ 0 \end{bmatrix}\right)^\top, \tag{2.1}$$

where Θ' are pre-defined locations on the rectified image I^r, acting as constants and $\Delta_{\Theta'}^{-1}$ is a constant matrix that depends only on Θ'. Let $\mathcal{P} = \{p_i\}_{i=1,...,N}$ be a grid of pixels on I, then $p_i = \mathcal{T}\hat{p}_i^r$ describes the relationship between pixels on I and I^r, where $\hat{p}_i^r$ is a uniform grid on I^r concatenated with some constants. Lastly, the rectified image is computed from the grid $\mathcal{P}$ via a bilinear sampler [15]:

$$I_{ci}^r = \sum_{n,m}^{H,W} I_{cnm} f(1 - |p_{ix} - m|) f(1 - |p_{iy} - n|), \tag{2.2}$$

where, for each channel c, I^r_{ci} is the i-th pixel on I^r, I_{cnm} is pixel (n, m) on I, p_{ix} and p_{iy} are the x and y coordinates of the grid map p_i, and $f(\cdot)$ denotes the ReLU function.

2.2.2 Connectionist Temporal Classification Decoder

The CTC decoder uses an additional *blank* token in $\mathcal{C}$. Denote the set of all possible sequences of predicted classes with length T as $\mathcal{C}^T$, and let $\mathcal{B}$ be a many-to-one mapping which maps $\pi \in \mathcal{C}^T$ to the decoded text ℓ by removing repeated predictions and *blank* tokens. For instance, $\mathcal{B}(\texttt{-ff-l-yy--}) = \texttt{fly}$ when $T = 10$, where `-` denotes the *blank* token. Given an input I, the *conditional probability* for ℓ conditioned on the network output $\mathcal{N}(I)$ is defined as a sum over the preimage of ℓ under $\mathcal{B}$:

$$\Pr(\ell \mid \mathcal{N}(I)) = \sum_{\pi \in \mathcal{B}^{-1}(\ell)} \Pr(\pi \mid \mathcal{N}(I)), \tag{2.3}$$

where $\Pr(\pi \mid \mathcal{N}(I)) = \prod_{t=1}^{T} \mathcal{N}_{t,\pi_t}(I)$ is the product over probabilities of predicting label π_t at frame t. The optimal decoded text ℓ^* is the maximum likelihood solution of 2.3, and we follow Graves *et al.* [12] to approximate the maximum likelihood solution by

$$\ell^* \approx \mathcal{B}(\arg\max_{\pi}(\pi \mid \mathcal{N}(I))), \tag{2.4}$$

meaning the most likely label for each frame is selected before applying $\mathcal{B}$ to recover ℓ^*. The CTC decoder $\mathcal{D}$ is then defined as 2.4, where $\mathcal{D}(\mathcal{N}(I)) = \ell^*$.

2.2.3 Attention Decoder

The attention decoder relies on an LSTM to output the sequence of T labels, where the inputs to each LSTM cell $t' \in [1..T]$ are the attention weighted average over $\mathcal{H}$, the previous predicted label and the previous hidden state from cell $t'-1$, as shown in 2b. Let $\mathcal{N}(I) = \mathcal{H} = [h_1, ..., h_T]$ be the contextual features. For each $t' \in [1..T]$, denote $s_{t'}$ as the hidden state output of the t'-th LSTM cell, and then $y_{t'} = \sigma_{soft}(W_0 x_{t'} + b_0)$ is the class prediction output for frame t' given the hidden states, where W_0 and b_0 are trainable parameters and σ_{soft} is the Softmax function.

To obtain the attention weights for the inputs to cell t', first define $e_{t',t} = a^\top \tanh(W s_{t'-1} + V h_t + b)$, where W, V, a, b are trainable parameters. Then, define

$$\alpha_{t',t} = \sigma_{soft}(e_{t',\cdot}) = \frac{\exp(e_{t',t})}{\sum_{j=1}^{T} \exp(e_{t',j})} \tag{2.5}$$

to be the attention weights. The attention operation then linearly combines contextual features $[h_1, ..., h_T]$ using the attention weights to obtain $g_{t'} = \sum_{t=1}^{T} \alpha_{t',t} \cdot h_t$. The LSTM cell of the decoder is then recurrently updated by

$$s_{t'} = \text{LSTM}([g_{t'}, h(y_{t'-1})], s_{t'-1}), \tag{2.6}$$

where $y_{t'-1}$ is the previous predicted label and $h(\cdot)$ is the one-hot embedding. This yields a sequence of predicted labels $y_{t'}$, in which the *end-of-sentence* token [s] indicates the end of the output word.

2.3 Vision Transformers for STR

ViTSTR [2] is an adaptation of the standard Vision Transformer (ViT [8]) to the STR task. ViT adopts the Transformer [46] encoder originally designed for NLP tasks, where ViTSTR further extends ViT by modifying the prediction head to predict an ordered sequence of labels, instead of a single label, for classification.

The general architecture of ViTSTR is given in 2c. The input image $I \in \mathbb{R}^{H \times W \times C}$ is cut into a sequence of T patches shaped $P \times P \times C$. It is then flattened and converted via a linear projection to what we refer to as the sequence patch embedding. An extra learnable class embedding is prepended to the sequence of patch embeddings, where unique positional encodings are added to each patch embedding. The resulting patch embeddings are the input to the Transformer encoder, where T features are extracted from the encoder instead of just one. A prediction head then projects these features into T label predictions. This sequence of labels always start with an *begin-of-sentence* token [GO], whereas the *end-of-sentence* token [s] indicates the end of the output word. Unlike LSTM-based models, Transformers can predict the sequence of labels in parallel.

3 STR-Cert

We introduce STR-Cert, a polyhedral verification method based on DeepPoly [43], and propose novel algorithms and polyhedral bounds that are necessary to certify STR models.

3.1 Polyhedral Verification

While several robustness certification methods have been proposed, as overviewed in 1, STR-Cert utilizes polyhedral verification for its balance of scalability

and precision, offering a sweet spot for STR certification. Specifically, we adopt the DeepPoly abstract domain [43], which is a sub-polyhedral abstract domain that maintains lower and upper *polyhedral bounds* and *interval bounds* for each neuron. Formally, let $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ be an ordered set of neurons in $\mathcal{N}$ such that the order complies with the order of the layers they belong to. For each neuron x_j, we define the interval bounds $l_j \leq x_j \leq u_j$ and the polyhedral bounds $\sum_{i<j} a_i^l \cdot x_i + b^l \leq x_j \leq \sum_{i<j} a_i^u \cdot x_i + b^u$, where $l_j, u_j, a_i^l, b^l, a_i^u, b^u \in \mathbb{R} \cup \{-\infty, \infty\}$. To certify robustness, DeepPoly adopts *backsubstitution* to bound the difference between neurons in the network logits output, i.e., whether $\mathcal{N}_{t,c}(I') - \mathcal{N}_{t,c'}(I') > 0 \quad \forall I' \in \mathbb{S}_I$, to certify no class change occurs under perturbation. It recursively substitutes target neurons with the polyhedral bounds of previous layers' neurons until reaching the input neurons. We note that DeepPoly is algorithmically equivalent to CROWN [58], and we refer the reader to [43] for details of DeepPoly.

The polyhedral verification framework has been adopted to certifying FNCCs [43], CNNs [6], LSTMs [36], and the Softmax function [50]. To certify STR models introduced in 2.2, we first derive novel polyhedral bounds for the network components not covered in the literature, namely TPS, patch embedding and positional encoding in 3.2 and 3.3, while other relevant polyhedral bounds are provided in A. Secondly, we bridge the gap between neural network predicted sequence of labels and the final predicted text. For attention decoder and ViTSTR models, the sequence of labels are directly used as the predicted text, meaning any change of label before the *end-of-sentence* token guarantees a change in the predicted text. CTC decoder models, however, operate differently and we provide an algorithm to certify them in 3.4. Finally, since Softmax is a key component in attention decoder and ViTSTR, we propose novel polyhedral bounds for Softmax in 3.5 that refines existing bounds by considering the constraint that Softmax outputs sum to 1.

3.2 TPS transformation

Recall from 2.2.1 that the TPS transformation utilizes a localization CNN to produce the fiducial points Θ, before computing matrix $\mathcal{T}$ and generating the grid of pixels $\mathcal{P}$ on I, where $p_i = \mathcal{T}\hat{p}_i^r$. When I is under perturbation $\mathbb{S}_I$, polyhedral bounds for Θ can be computed by existing DeepPoly methods for CNNs, which can be directly applied to bound $\mathcal{T}$. Since $\hat{p}_i^r$ is a uniform grid on I^r that are unaffected by perturbations to I, $\mathcal{P}$ is a linear transformation from $\mathcal{T}$, which can be adopted as the polyhedral bounds for $\mathcal{P}$.

To derive polyhedral bounds for the rectified image I^r, recall the bilinear map, where $f(\cdot)$ is the ReLU function:

$$I^r_{ci} = \sum_{n,m}^{H,W} I_{cnm} f(1 - |p_{ix} - m|) f(1 - |p_{iy} - n|). \tag{3.1}$$

Since p_{ix}, p_{iy} and I_{cnm} are all under perturbation, we first derive polyhedral bounds for $r_{ix} \coloneqq f(1 - |p_{ix} - m|)$ from interval bounds $l_{ix} < p_{ix} < u_{ix}$. The function mapping p_{ix} to r_{ix} consists of four pieces of linear functions as shown in 3. Assume WLOG that $m = 0$, then if l_{ix}, u_{ix} are within the left two or right two pieces of linear functions, i.e., $l_{ix}, u_{ix} \in [-\infty, 0]$ or $[0, \infty]$, the polyhedral bounds follows from ReLU (see A.2). If $l_{ix} \in [-1, 0]$ and $u_{ix} \in [0, 1]$, the polyhedral bounds are illustrated in 3a:

$$\frac{-(u_{ix} + l_{ix})}{u_{ix} - l_{ix}} p_{ix} + \frac{2u_{ix}l_{ix}}{u_{ix} - l_{ix}} + 1 \leq r_{ix} \leq a^u_1 p_{ix} + 1, \tag{3.2}$$

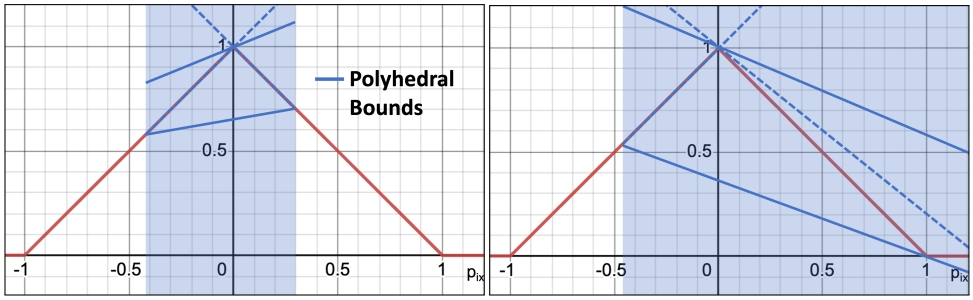
where $a^u_1 \in [-1, 1]$. If l_{ix}, u_{ix} are across three linear functions, first consider the case of 3b. The bounds are

$$\frac{-(1 + l_{ix})}{1 - l_{ix}} p_{ix} + \frac{2l_{ix}}{1 - l_{ix}} + 1 \leq r_{ix} \leq a^u_2 p_{ix} + 1, \tag{3.3}$$

where $a^u_2 \in [-1/u_{ix}, 1]$. The bounds for the other case of $l_{ix} \in [-\infty, -1], u_{ix} \in [0, 1]$ follow from symmetry. Lastly, if l_{ix}, u_{ix} are across all four linear functions, the bounds are $0 \leq r_{ix} \leq a^u_3 p_{ix} + 1$, where $a^u_3 \in [-1/u_{ix}, -1/l_{ix}]$. Bounds for $r_{iy} \coloneqq f(1 - |p_{iy} - n|)$ can be derived similarly. In practice, a^u_1, a^u_2 and a^u_3 are chosen such that the area bounded by the interval and polyhedral bounds is minimized. Detailed bounds for all cases are provided in A.1. With the polyhedral bounds of I_{cnm}, r_{ix} and r_{iy}, the final polyhedral bounds for I^r can be computed via addition and multiplication bounds (see A.3).

3.3 Patch Embedding and Positional Encoding

Patch embedding and positional encoding are two crucial components of ViTSTR as introduced in 2.3. Recall that patch embedding produces a sequence of T patches before flattened into vectors of dimension P^2C and linearly projected into a embedding of size D. The composition of patching $\phi_1 : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^{N \times P^2 C}$ and linear projection $\phi_2 : \mathbb{R}^{N \times P^2 C} \to \mathbb{R}^{N \times D}$ is equivalent to a convolution operation with kernel size $(P \times P)$, stride



(a) $l_{ix} \in [-1, 0], u_{ix} \in [0, 1]$ (b) $l_{ix} \in [-1, 0], u_{ix} \in [1, \infty]$

Figure 3: Polyhedral bounds for $f(1 - |p_{ix} - m|)$ against p_{ix} in the bilinear map of TPS.

size P, input channels C and output channels D. The kernel weights is obtained by reshaping the linear projection weights from shape $(P^2C \times D)$ to $(D \times C \times P \times P)$. The patch embedding can thus be certified as a convolution layer.

We denote the outputs from the patch embedding as $v^p \in \mathbb{R}^{N \times D}$. A unique positional encoding of the same dimension D is added to each patch (for details see [2]). Let $e^p \in \mathbb{R}^{N \times D}$ be the positional encoding matrix for the whole image, then the positional encoding layer corresponds to $v^p + e^p$. Since e^p is constant with respect to perturbations, positional encoding can be certified as a linear layer with the identify matrix as weight and e^p as bias.

3.4 CTC Decoder Certification

Recall that $\mathcal{N}_{t,c}(I)$ is the logits output for frame t and class c. To certify the output sequence of the CTC decoder $\mathcal{D}$, we need to check that, for the perturbed input domain $\mathbb{S}_I$, all the combinations of possible predictions from each frame t still produce the ground truth sequence after applying reduction mapping $\mathcal{B}$. Algorithm 1 outlines the procedure for CTC decoder certification. At line 6, $V(\mathbb{S}_I, \mathcal{N}_{t,l_t} > \mathcal{N}_{t,c})$ returns unsafe if these exist $I' \in \mathbb{S}_I$ such that $\mathcal{N}_{t,l_t}(I') \leq \mathcal{N}_{t,c}(I')$. Therefore, M_t is the set of all possible classifications for frame t under perturbation. To justify line 8-9, if there exists t such that $|M_t| > 3$, let l_t be the true prediction label for frame t. Then, there must exist $c_t \in M_t$ such that $c_t \neq l_{t-1}, l_t$ and l_{t+1} by the pigeon hole principle. This means c_t will not be a repeated label and $\mathcal{B}$ will produce a different sequence to the ground truth, failing the certification. If

Algorithm 1: CTC Decoder Certification

Input: CTC decoder $\mathcal{D}$, input I, network $\mathcal{N}$, adversarial region $\mathbb{S}_I$, verifier V

1: $\ell^* \leftarrow \mathcal{D}(\mathcal{N}(I))$
2: **for** $t \leftarrow T$ **do**
3: $\quad l_t \leftarrow \text{argmax}_k(\mathcal{N}_{t,k}(I))$
4: $\quad M_t \leftarrow [l_t]$
5: $\quad$ **for** $c \leftarrow \mathcal{C}$ **do**
6: $\quad\quad$ **if** $V(\mathbb{S}_I, \mathcal{N}_{t,l_t} > \mathcal{N}_{t,c})$=*unsafe*, $l_t \neq c$ **then**
7: $\quad\quad\quad M_t.append(c)$
8: $\quad$ **if** $|M_t| > 3$ **then**
9: $\quad\quad$ **return** Unsafe
10: **for** $\pi \leftarrow (M_1 \times M_2 \times ... \times M_T)$ **do**
11: $\quad$ **if** $\mathcal{B}(\pi) \neq \ell^*$ **then**
12: $\quad\quad$ **return** Unsafe
13: **return** Safe

$|M_t| \leq 3$ for all t, we check all possible sequences $\pi \in (M_1 \times M_2 \times ... \times M_T)$ for text changes after applying $\mathcal{B}$ at line 10-12. If all possible sequences are safe, the CTC decoder model is then certified to be safe under $\mathbb{S}_I$. Note that the worst-case complexity for 1 is $\mathcal{O}(3^{|T|})$, but without line 8-9, it will increase to $\mathcal{O}(|\mathcal{C}^T|)$, where often $|\mathcal{C}| > 40$. However, we find that, in practice, $|M_t|$ increases monotonically with t, and usually quite dramatically because LSTM certification loses precision as the network is unfolded. This means Algorithm 1 typically terminates at line 9, and if not, the search space of $(M_1 \times M_2 \times ... \times M_T)$ is in the order of 4!, which is constant time.

3.5 Refining Softmax Bounds

Softmax is an important function in both ViTSTR and the attention decoder. Existing Softmax polyhedral bounds [50, 42] only consider element-wise bounds between the inputs and outputs of Softmax, and they fail to consider the crucial constraint that the Softmax output sum up to 1. Let the Softmax output neurons x_i, $i \in [1..N]$ have interval bounds $l_i \leq x_i \leq u_i$, respectively. We aim to refine Softmax by introducing a novel polyhedral transformation of the existing Softmax polyhedral bounds that incorporates the additional constraint $\sum_{i=1}^{N} x_i = 1$. This constraint unfortunately cannot

be described by a linear combination of previous neurons. Instead of over-approximating lower and upper polyhedral bounds, we find the minimum relaxation to the constraints such that a linear transformation can exactly describe it. By removing the interval bound constraint for a single neuron x_k, the following constraints

$$l_i \leq x_i^r \leq u_i \quad \forall i \neq k \quad \text{where} \sum_{i=1}^{N} x_i^r = 1, \tag{3.4}$$

can be exactly satisfied by the refined neurons x_i^r:

$$x_i^r = x_i \text{ for } i \neq k \text{ and } x_k^r = 1 - \sum_{i \neq k} x_i. \tag{3.5}$$

In this case, the polyhedral bounds are this linear transformation, making the polyhedral transformation exact. Next, we discuss how to choose k such that the constraint of 3.4 is the tightest. From 3.5 and 3.4, we can deduce implied upper and lower bounds for x_k^r:

$$1 - \sum_{i \neq k} u_i \leq x_k^r = 1 - \sum_{i \neq k} x_i =\leq 1 - \sum_{i \neq k} l_i, \tag{3.6}$$

where the tightness of these implied bounds for x_k^r is

$$(1 - \sum_{i \neq k} l_i) - (1 - \sum_{i \neq k} u_i) = \sum_{i \neq k} (u_i - l_i). \tag{3.7}$$

Therefore, by choosing $k = \text{argmax}_i(u_i - l_i)$ with the loosest bounds, the implied bounds for x_k^r will be the tightest, ensuring it is a minimum relaxation from the original constraints. Note that, when $1 - \sum_{i \neq k} l_i \leq u_k$ and $1 - \sum_{i \neq k} u_i \geq l_k$, the inequality $l_k \leq x_k^r \leq u_k$ can be inferred from 3.6 and we can impose the $\sum_{i=1}^{N} x_i = 1$ constraint without any relaxation.

4 Experiments

We present STR-Cert[1] by implementing 3 and extending DeepPoly [43]. We also adopt Prover [36] for certifying LSTMs and the element-wise Softmax bounds derived by Wei *et al.* [50]. We evaluate the performance and precision of STR-Cert on a range of STR networks and datasets, while providing insights into the comparisons between STR architectures and connections between adversarial training, prediction confidence, and robustness.

[1] Will be made open-source with the final version of this paper.

Model	CTC decoder			Attention decoder			ViTSTR			
Datasets	$\epsilon = .001$	$\epsilon = .003$	$\epsilon = .005$	$\epsilon = .001$	$\epsilon = .003$	$\epsilon = .005$	$\epsilon = .001$	$\epsilon = .003$	$\epsilon = .005$	$\epsilon = .01$
IIIT5K	98.5%	76.5%	48.5%	91.0%	68.0%	39.0%	97.5%	75.0%	57.5%	24.0%
IC13	99.5%	89.5%	59.0%	95.0%	82.0%	52.0%	98.5%	87.0%	74.5%	41.5%
IC15	95.5%	56.5%	18.0%	89.5%	35.0%	11.0%	95.5%	58.5%	41.0%	11.5%
SVT	95.0%	68.0%	34.0%	88.0%	42.5%	19.0%	97.5%	65.5%	58.5%	29.5%
SVTP	94.5%	62.5%	36.5%	88.5%	58.0%	21.5%	96.0%	61.5%	43.0%	21.5%
CUTE	97.5%	83.0%	41.0%	92.0%	73.0%	33.0%	96.5%	79.5%	58.0%	27.0%

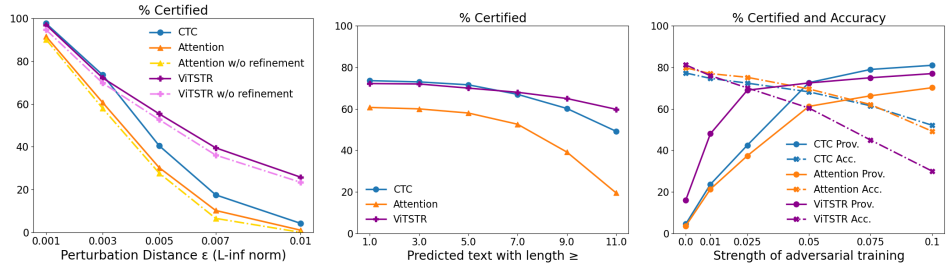
Table 1: % certified in the first 200 correctly classified instances for CTC decoder, attention decoder and ViTSTR model for 6 datasets.

4.1 Datasets, Models and Training

We adopt the training setup of Baek *et al.* [4] with the PyTorch [32] library. For all architectures, the models are trained for 100000 iterations on the synthetic MJSynth [14] and SynthText [13] datasets and validated on the training datasets of IIIT5K [27], IC13 Born-Digital Images [18], IC15 Focused Scene Text [17], SVT [47], SVTP [33] and CUTE80 [35]. We use AdamW [20] optimizer with a cosine decay scheduler, while also deploying PGD adversarial training [26], discussed further in 4.3. The attention decoder model utilizes TPS [40] as the transformation module, a 5 layer CNN-ReLU network as the feature extractor, an LSTM as the sequence model, and the attention decoder [52]. The CTC decoder model uses the same transformation, feature extractor, and sequence model architectures. For the Vision Transformer model, we use the architecture in ViTSTR [2] with 5 layers. The pre-trained ViTSTR models [2] are unfortunately too large and beyond the scope of this work. The CTC decoder and the attention decoder models both have around 500K parameters, whereas the ViTSTR have around 700K parameters. Details on architectures and training can be found in B.

4.2 Robustness Certification

We now provide results on the certified robustness for the CTC decoder, attention decoder, and ViTSTR model on the test datasets of IIIT5K [27], IC13 [18], IC15 [17], SVT [47], SVTP [33] and CUTE80 [35]. For each model, we examine the percentage of correctly classified samples that can be certified to remain correctly classified under perturbation, which we refer to as *percentage certified*. We certify the first 200 correctly classified samples in each dataset for varying perturbation budget, i.e., the maximum perturbation distance, of $\epsilon = 0.001$ up to 0.01 under L_∞ norm. The av-



(a) % certified against perturbation distances.

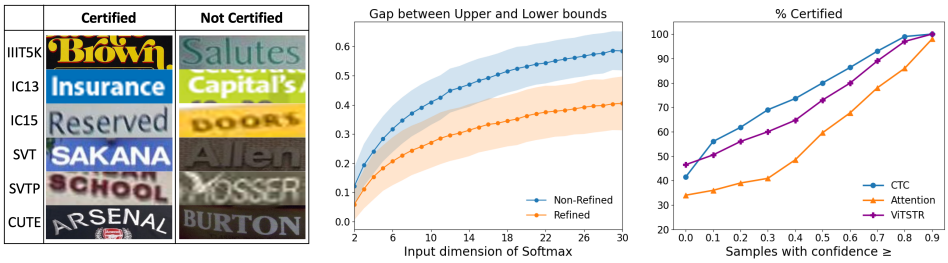
(b) % certified for samples with predicted text longer then various lengths.

(c) % certified and accuracy against the strength of PGD adversarial training.

Figure 4: Certification results with analysis against text length and adversarial training strength.

erage percentage certified for all architectures with and without Softmax refinement (in 3.5) are illustrated against perturbation budgets in 4a, and detailed results for certification with Softmax refinement are shown in 1. Note that the CTC decoder model does not have a Softmax layer, and an ablation study for Softmax refinement is provided in 4.4. We observe that the percentage certified for the CTC and attention decoder models drops more steeply for increasing perturbation budget ϵ than for the ViTSTR model. This is because each LSTM recurrent cell involves multiplication of two non-linear activated neurons, which is difficult to certify with high precision. Moreover, existing certifiers for LSTMs unfold the recurrent operation, which significantly increases the depth of the network on which the loss of precision compounds. In addition, harder datasets such as IC15 are consistently less robust across all model architectures and perturbation budgets. Fig. 5a includes examples of certified and uncertified samples, and additional certification results for STR models with different number of layers are provided in C, where similar trends can be observed.

Scalability with respect to the length of predicted text is particularly interesting. CTC and attention decoder models rely on LSTMs to recurrently predict the sequence of labels, which means the loss of precision compounds as the length of text increases. In practice, as shown in 4b, percentage certified drops significantly for the two LSTM-based models as the predicted text grow longer then 9, especially for the attention-decoder model, which contains two layers of LSTM and struggles to certify predicted text with length 11 or more. ViTSTR, however, shows a smaller drop in percentage



(a) Examples from each dataset that are certified and not certified against adversarial attacks.

(b) Tightness of bounds with and without Softmax refinement against input dimensions.

(c) % certified for samples with prediction confidence above various thresholds.

Figure 5: Example images and certification results for Softmax bounds and prediction confidence.

certified because it predicts in parallel, where longer text does not increase the depth of certification.

The average certification runtime per sample is 49s for CTC decoder model; 92s for attention decoder model; and 14s for ViTSTR. The discrepancy between runtime is mainly due to the certification of LSTM layers, as the CTC decoder and attention decoder models include one and two LSTM layers, respectively. If we take into consideration that large-scale LSTM-based STR models in practice usually include multiple layers of Bi-LSTM, it becomes extremely challenging, if not infeasible, to certify them with current methods. Vision Transformers, however, seem to be a better choice in terms of certification scalability.

4.3 Effect of Adversarial Training

To investigate the effect of adversarial training on percentage certified, we train various models in all three STR architectures with different strengths of adversarial training, i.e., the maximum perturbation budget in PGD [26], with 10 steps. For CTC and attention decoder models, standard PGD adversarial training is adopted to train models from scratch. For the ViTSTR model, we follow the adversarial training setup of ViT [28] and naturally train the model for 20000 epochs before training adversarially, which improves the training stability and adversarial accuracy.

The average percentage certified and accuracy across the test datasets for STR architectures are shown in 4c, where the adversarial training strength

ranges from 0 (natural training) to 0.1 and the samples are certified against perturbation budget $\epsilon = 0.003$. We observe that, as expected, there exists a trade-off between the accuracy of the model and the percentage certified. However, it is interesting to note that the optimal trade-off points vary for the different architectures. For CTC and attention decoder models, as the PGD strength rises above 0.05, the accuracy begins to drop markedly whilst the percentage certified only increases marginally. For ViTSTR, the optimal strength is, however, around 0.025 since the percentage certified stagnates with higher strength while the accuracy plummets. Nevertheless, the experiments confirm that adversarial training significantly boosts percentage certified.

4.4 Ablation on Softmax Refinement

To demonstrate effectiveness of our Softmax refinement (3.5), we first directly compare the tightness of the output neuron's bounds with and without Softmax refinement on a synthetic neural network. A feed-forward network with 2 hidden layers and ReLU activations, followed by the Softmax layer and a fully-connected layer that outputs a single scalar is used. We compare the gap between the upper and lower bounds of the final output neuron under $\epsilon = 0.1$ perturbation, with varying input dimensions to the Softmax function from 2 to 30. For each input dimension, we certify 1000 models with random parameters for the final layer and random inputs. The mean plus and minus one standard deviation of the gaps between bounds are shown in 5b, where the Softmax refinement provides considerably tighter bounds, even when the input dimensions for Softmax are high, as in the case of Vision Transformers.

On large-scale STR models, we demonstrate the improvement to percentage certified from Softmax refinement in 4a. This improvement is present in all datasets for the attention decoder model and ViTSTR. In practice, we never observe an instance that the Softmax refinement harms the percentage certified.

4.5 Prediction Confidence and Certification

Since polyhedral verification provides bounds on the logits output of the network, samples with high prediction confidence, i.e., the product of confidence over the sequence of predictions, should be easier to certify. In 5c, percentage certified for samples with prediction confidence above various

thresholds are plotted. When the lower confidence samples are filtered out, percentage certified dramatically increases, up to near 100% when only high confidence samples remain. For the CTC decoder model, there are certain frames t that can predict different labels without changing the decoded text. Those frames usually have lower prediction confidence, but this does not necessarily imply a lack of robustness. This is evident in 5c since the percentage certified for the CTC decoder model actually exceeds that of other architectures when the confidence threshold ≥ 0.1.

5 Conclusion

We developed STR-Cert, the first robustness certification method for STR models and Vision Transformers, with novel algorithms and polyhedral bounds. We certified and compared three STR model architectures, where we demonstrated scalability issues of LSTM-based models and elucidated the benefits of Vision Transformers. Future work includes studying the robustness of pre-trained ViTSTR models in relation to its training dataset, extending the method to incorporate branch and bound, GPU parallelization, and certifying perturbations in other L_p norms.

Acknowledgments

This project received funding from the ERC under the European Union's Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No. 834115) and ELSA: European Lighthouse on Secure and Safe AI project (grant agreement No. 101070617 under UK guarantee).

References

[1] Michael E Akintunde, Andreea Kevorchian, Alessio Lomuscio, and Edoardo Pirovano. Verification of RNN-based neural agent-environment systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6006–6013, 2019. 2

[2] Rowel Atienza. Vision transformer for fast and efficient scene text recognition. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 12821 LNCS:319–334, 5 2021. 2, 6, 8, 11, 14

[3] E Bacci, M Giacobbe, and D Parker. Verifying reinforcement learning up to infinity. *Proceedings of the International Joint Conference on Artificial Intelligence*, 2021. 4

[4] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-October:4714–4722, 4 2019. 6, 14, 38

[5] Gregory Bonaert ETH Zurich, Dimitar I Dimitrov ETH Zurich, Maximilian Baader ETH Zurich, and Martin Vechev ETH Zurich. Fast and precise certification of transformers. *International Conference on Programming Language Design and Implementation*, 2021. 2, 3

[6] Akhilan Boopathy, Tsui-Wei Weng, Pin-Yu Chen, Sijia Liu, and Luca Daniel. CNN-Cert: An efficient framework for certifying robustness of convolutional neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:3240–3247, 2019. 4, 9

[7] Jeremy Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:2323–2356, 2 2019. 1

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR 2021 - 9th International Conference on Learning Representations*, 10 2021. 8

[9] Krishnamurthy Dvijotham, Robert Stanforth, Sven Gowal, Timothy A Mann, and Pushmeet Kohli. A dual approach to scalable verification of deep networks. In *UAI*, volume 1, page 3, 2018. 3

[10] Marc Fischer, Maximilian Baader, and Martin Vechev. Scalable certified segmentation via randomized smoothing. *Proceedings of the International Conference on Machine Learning*, 139:3340–3351, 7 2021. 4

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR*, 2015. 1

[12] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ACM International Conference Proceeding Series*, volume 148, pages 369–376, 2006. 6, 7

[13] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. Synthetic data for text localisation in natural images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016-December:2315–2324, 4 2016. 14, 36

[14] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *In Workshop on Deep Learning, NIPS*, 6 2014. 14, 36

[15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *Advances in Neural Information Processing Systems*, 2015-January:2017–2025, 6 2015. 6

[16] Matt Jordan and Alexandros G Dimakis. Exactly computing the local Lipschitz constant of ReLU networks. *Conference on Neural Information Processing Systems*, 2020. 3

[17] Dimosthenis Karatzas, Lluis Gomez-Bigorda, Anguelos Nicolaou, Suman Ghosh, Andrew Bagdanov, Masakazu Iwamura, Jiri Matas, Lukas Neumann, Vijay Ramaseshan Chandrasekhar, Shijian Lu, Faisal Shafait, Seiichi Uchida, and Ernest Valveny. ICDAR 2015 competition on robust reading. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 2015-November:1156–1160, 11 2015. 14, 37

[18] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez I. Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere De Las Heras. ICDAR 2013 robust reading competition. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 2013. 14, 37

[19] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. *Computer Aided Verification*, 10426 LNCS:97–117, 2017. 3

[20] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015. 14, 38

[21] Ching Yun Ko, Zhaoyang Lyu, Tsui Wei Weng, Luca Daniel, Ngai Wong, and Dahua Lin. POPQORN: Quantifying robustness of recurrent neural networks. In *36th International Conference on Machine Learning, ICML*, volume 2019-June, pages 6031–6087, 2019. 2, 4

[22] Linyi Li, Tao Xie, and Bo Li. SoK: Certified robustness for deep neural networks. *Proceedings - IEEE Symposium on Security and Privacy*, 2023-May:1289–1310, 9 2023. 2, 4

[23] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018. 2

[24] Hsuan-Cheng Liao, Chih-Hong Cheng, Maximilian Kneissl, and Alois Knoll. Are attention networks more robust? towards exact robustness verification for attention networks. In *Computer Safety, Reliability, and Security: 41st International Conference*, 2 2022. 2

[25] Wei Liu, Chaofeng Chen, Kwan-Yee K Wong, Zhizhong Su, and Junyu Han. STAR-Net: A spatial attention residue network for scene text recognition. *The British Machine Vision Conference*, 2016. 6

[26] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *6th International Conference on Learning Representations*, 6 2018. 14, 16, 38

[27] Anand Mishra, Karteek Alahari, and C V Jawahar. Scene text recognition using higher order language priors. *The British Machine Vision Association*, 2012. 14, 37

[28] Yichuan Mo, Dongxian Wu, Yifei Wang, Yiwen Guo, and Yisen Wang. When adversarial training meets vision transformers: Recipes from training to architecture. *Advances in Neural Information Processing Systems*, 35, 10 2022. 16

[29] Christoph Müller, François Serre, Gagandeep Singh, Markus Püschel, and Martin Vechev. Scaling polyhedral neural network verification on gpus. *Proceedings of the 4 th MLSys Conference*, 7 2021. 4

[30] Mark Niklas Müller, Gleb Makarchuk, Gagandeep Singh, Markus Püschel, and Martin Vechev. PRIMA: General and precise neural network certification via scalable convex hull approximations. *Proceedings of the ACM on Programming Languages*, 6, 3 2021. 3

[31] Raphael Olivier and Bhiksha Raj. Sequential randomized smoothing for adversarially robust speech recognition. *EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 6372–6386, 11 2021. 4

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 12 2019. 14, 38

[33] Trung Quy Phan, Palaiahnakote Shivakumara, Shangxuan Tian, and Chew Lim Tan. Recognizing text with perspective distortion in natural scenes. *Proceedings of the IEEE International Conference on Computer Vision*, pages 569–576, 2013. 14, 37

[34] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. *Advances in Neural Information Processing Systems*, 2018-December:10877–10887, 11 2018. 3

[35] Anhar Risnumawan, Palaiahankote Shivakumara, Chee Seng Chan, and Chew Lim Tan. A robust arbitrary text detection system for natural scene images. *Expert Systems with Applications*, 41:8027–8048, 12 2014. 14, 38

[36] Wonryong Ryou, Jiayu Chen, Mislav Balunovic, Gagandeep Singh, Andrei Dan, and Martin Vechev. Scalable polyhedral verification of recurrent neural networks. *Computer Aided Verification*, 12759 LNCS:225–248, 2020. 2, 3, 4, 9, 13, 29, 31, 32, 33

[37] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[38] Kevin Scaman and Aladin Virmaux. Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 3835–3844. Neural information processing systems foundation, 2018. 3

[39] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2298–2304, 2017. 2, 6

[40] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:4168–4176, 3 2016. 6, 14

[41] Zhouxing Shi, Qirui Jin, Huan Zhang, Zico Kolter, Suman Jana, and Cho-Jui Hsieh. Formal verification for neural networks with general nonlinearities via branch-and-bound. *The second Workshop on Formal Verification of Machine Learning, ICML*, 2023. 4

[42] Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. Robustness verification for transformers. *Proceedings of the International Conference on Learning Representations*, 2 2020. 2, 4, 12, 28, 32

[43] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–30, 2019. 3, 4, 8, 9, 13, 26, 28, 31, 39, 40

[44] Marco Sälzer and Martin Lange. Reachability is np-complete even for the simplest neural networks. *International Conference on Reachability Problems*, 13035 LNCS:149–164, 8 2021. 3

[45] Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating Robustness of Neural Networks with Mixed Integer Programming. *7th International Conference on Learning Representations, ICLR 2019*, 11 2017. 2, 3

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 8

[47] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1457–1464, 2011. 14, 37

[48] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Efficient formal safety analysis of neural networks. *Advances in neural information processing systems*, 31, 2018. 3

[49] Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network nobustness verification. *Conference on Neural Information Processing Systems*, 3 2021. 3, 4

[50] Dennis Wei, Haoze Wu, Min Wu, Pin-Yu Chen, Clark Barrett, and Eitan Farchi. Convex bounds on the Softmax function with applications to robustness verification. *International Conference on Artificial Intelligence and Statistics*, 2023. 9, 12, 13, 32, 34

[51] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5276–5285. PMLR, 2018. 3

[52] Zbigniew Wojna, Alexander N. Gorban, Dar Shyang Lee, Kevin Murphy, Qian Yu, Yeqing Li, and Julian Ibarz. Attention-based extraction of structured information from street view imagery. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 2017. 2, 6, 14

[53] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International conference on machine learning*, pages 5286–5295. PMLR, 2018. 3

[54] Min Wu and Marta Kwiatkowska. Robustness guarantees for deep neural networks on videos. *Proceedings of the IEEE*, 2020. 2

[55] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 2020-December, 2 2020. 1

[56] Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. *9th International Conference on Learning Representations*, 11 2021. 3, 4

[57] Xiaoyong Yuan, Pan He, Xiaolin Lit, and Dapeng Wu. Adaptive adversarial attack on scene text recognition. *Conference on Computer Communications Workshops*, pages 358–363, 7 2020. 2

[58] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems*, volume 2018-, pages 4939–4948. Neural information processing systems foundation, 2018. 1, 2, 3, 4, 9

[59] Huan Zhang, Pengchuan Zhang, and Cho-Jui Hsieh. Recurjac: An efficient recursive algorithm for bounding jacobian matrix of neural networks and its applications. *Proceedings of the Thirty-Third AAAI Conference*, 2019. 3

[60] Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing. *ACM Transactions on Intelligent Systems and Technology*, 11(3):1–41, 2020. 1

Part I
Appendix

Contents

A Background on Polyhedral Verification Bounds

In this section, for the sake of completeness we provide polyhedral bounds for the relevant layers in OCR/STR networks. While the bounds for these layers, apart from the STN bilinear map, are mostly covered in the literature, we include them as background for the modifications and extensions we applied. Note that, since fully-connected (linear) layers can be transformed exactly, and CNN layers can be transformed into fully-connected layers [43], we only cover polyhedral bounds for non-linear layers.

A.1 STN Bilinear Map

Recall the bilinear map, where $f(\cdot)$ is the ReLU function:

$$I^r_{ci} = \sum_{n,m}^{H,W} I_{cnm} f(1 - |p_{ix} - m|) f(1 - |p_{iy} - n|). \tag{A.1}$$

Here, we provide detailed polyhedral bounds for $r_{ix} \coloneqq f(1 - |p_{ix} - m|)$ from interval bounds $l_{ix} < p_{ix} < u_{ix}$. Since the function mapping p_{ix} to r_{ix} consists of four pieces of linear functions, and l_{ix}, u_{ix} can be in either of the four pieces, we enumerate all possible cases here.

- If l_{ix}, u_{ix} are within the same linear piece, let ∇ be the gradient of the piece of linear function l_{ix} and u_{ix} are both inside ($\nabla = 0, 1 or -1$), then $\nabla \leq r_{ix} \leq \nabla$.
- If $[l_{ix}, u_{ix}]$ are across exactly two linear pieces, then:
  1. If $l_{ix} \in (-\infty, -1)$ and $u_{ix} \in [-1, 0]$, then $u_{ix}(p_{ix} + 1 - l_{ix})/(u_{ix} - l_{ix}) \leq r_{ix} \leq a_1 p_{ix} + a_1$ where $a_1 \in [0, 1]$.
  2. If $l_{ix} \in [0, 1)$ and $u_{ix} \in [1, \infty)$, then $u_{ix}(1 - p_{ix} - l_{ix})/(u_{ix} - l_{ix}) \leq r_{ix} \leq a_2 p_{ix} - a_2$ where $a_2 \in [-1, 0]$.
  3. If $l_{ix} \in [-1, 0)$ and $u_{ix} \in [0, 1]$, then

$$\frac{-(u_{ix} + l_{ix})}{u_{ix} - l_{ix}} p_{ix} + \frac{2u_{ix} l_{ix}}{u_{ix} - l_{ix}} + 1 \leq r_{ix} \leq a_3 p_{ix} + 1, \text{ where } a_3 \in [-1, 1]. \tag{A.2}$$

- If $[l_{ix}, u_{ix}]$ are across exactly three linear pieces, then:
  1. If $l_{ix} \in [-1, 0]$ and $u_{ix} \in [1, \infty)$, then

$$\frac{-(1 + l_{ix})}{1 - l_{ix}} p_{ix} + \frac{2l_{ix}}{1 - l_{ix}} + 1 \leq r_{ix} \leq a_4 p_{ix} + 1, \text{ where } a_4 \in [-1/u_{ix}, 1]. \tag{A.3}$$

  2. If $l_{ix} \in (-\infty, -1]$ and $u_{ix} \in [0, 1]$, then

$$\frac{-(u_{ix} - 1)}{u_{ix} + 1} p_{ix} + \frac{-2u_{ix}}{u_{ix} + 1} + 1 \leq r_{ix} \leq a_5 p_{ix} + 1, \text{ where } a_5 \in [-1, -1/l_{ix}]. \tag{A.4}$$

- Finally, if $[l_{ix}, u_{ix}]$ are across all four linear pieces, then $0 \leq r_{ix} \leq a^u_3 p_{ix} + 1$, where $a^u_3 \in [-1/u_{ix}, -1/l_{ix}]$.

This concludes all possible cases for l_{ix} and u_{ix}. Bounds for $r_{iy} \coloneqq f(1 - |p_{iy} - n|)$ can be derived similarly, and with the polyhedral bounds of I_{cnm}, r_{ix} and r_{iy}, the final polyhedral bounds for I^r can be computed via addition and multiplication bounds (A.3).

A.2 ReLU

To the best of our knowledge, polyhedral bounds for the ReLU activation function $\sigma(x) = \max\{x, 0\}$ were first introduced in [43]. Given an input $x \in [l_x, u_x]$ and with polyhedral upper and lower bounds $a^{\geq}(x), a^{\leq}(x)$, they distinguish the following three cases:

1. If $u_x \leq 0$, then $a^{\leq}(\sigma(x)) = a^{\geq}(\sigma(x)) = 0$ and $l_{\sigma(x)} = u_{\sigma(x)} = 0$.
2. If $0 \leq l_x$, then $a^{\geq}(\sigma(x)) = a^{\leq}(\sigma(x)) = x$, $l_{\sigma(x)} = l_x$ and $u_{\sigma(x)} = u_x$.
3. Otherwise, they set $a^{\geq}(\sigma(x)) = u_x(x - l_x)/(u_x - l_x)$ and $a^{\leq}(\sigma(x)) = \lambda x$ for $\lambda \in [0, 1]$. In practice, they choose the λ that minimizes the area between the upper and lower bounds in the $(x, \sigma(x))$-plane. Finally, they let $l_{\sigma(x)} = \lambda l_x$ and $u_{\sigma(x)} = u_x$.

A.3 Multiplication

To bound the product of two scalar variables $x \in [l_x, u_x]$ and $y \in [l_y, u_y]$ under perturbation, [42] use lower and upper polyhedral planes parameterized by coefficients A_l, B_l, C_l and A_u, B_u, C_u, i.e.

$$A_l x + B_l y + C_l \leq xy \leq A_u x + B_u y + C_u \tag{A.5}$$

They show that choosing $A_l = l_y, A_u = u_y, B_l = B_u = l_x, C_l = -l_x l_y$, and $C_u = -l_x u_y$ is optimal in that this choice of parameters minimizes the integrals of $F^L(x, y)$ and $F^U(x, y)$ over $[l_x, u_x] \times [l_y, u_y]$, where

$$\begin{aligned} F^L(x, y) &= xy - (A_l x + B_l y + C_l) \\ F^U(x, y) &= xy - (A_u x + B_u y + C_u) \end{aligned} \tag{A.6}$$

For the proof of this result, see [42].

A.4 LSTM

We adopt Prover [36] for certifying LSTM layers, and we briefly introduce their polyhedral bounds in this section. The idea behind the LSTM architecture is to handle long-term sequential dependencies, for example in words or sentences. These dependencies are passed through time with two vectors, a cell state $c^{(t)}$ and a hidden state $h^{(t)}$ for every timestep t. The state vectors are updated using the following equations:

$$\begin{aligned} f_0^{(t)} &= [x^{(t)}, h^{(t-1)}]W_f + b_f & i_0^{(t)} &= [x^{(t)}, h^{(t-1)}]W_i + b_i \\ o_0^{(t)} &= [x^{(t)}, h^{(t-1)}]W_o + b_o & \tilde{c}_0^{(t)} &= [x^{(t)}, h^{(t-1)}]W_{\tilde{c}} + b_{\tilde{c}} \\ c^{(t)} &= \sigma(f_0^{(t)}) \odot c^{(t-1)} + \sigma(i_0^{(t)}) \odot \tanh(\tilde{c}_0^{(t)}) & h^{(t)} &= \sigma(o_0^{(t)}) \odot \tanh(c^{(t)}) \end{aligned} \tag{A.7}$$

where $[\cdot, \cdot]$ denotes the horizontal concatenation of two row vectors, W and b denote the kernel and bias of the cell, and σ denotes the sigmoid function.

Polyhedral Verification of LSTM: [36] bound the products of the identity, sigmoid and tanh functions using lower and upper polyhedral planes parameterized by coefficients A_l, B_l, C_l and A_u, B_u, C_u. For

$$h(x, y) = \begin{cases} \sigma(x)\tanh(y) \\ \sigma(x)y \end{cases},$$

it follows that

$$A_l \cdot x + B_l \cdot y + C_l \leq h(x, y) \leq A_u \cdot x + B_u \cdot y + C_u \tag{A.8}$$

The problem of finding the bounding polyhedral planes can be reduced to an optimization problem. For the lower bound

$$\begin{aligned} &\min_{A_l, B_l, C_l} \int_{(x,y)\in B} (h(x, y) - (A_l \cdot x + B_l \cdot y + C_l)) \\ &\text{subject to } A_l \cdot x + B_l \cdot y + C_l \leq h(x, y), \quad \forall (x, y) \in B \end{aligned} \tag{A.9}$$

where $B = [l_x, u_x] \times [l_y, u_y]$ is the input boundary region of neurons x and y. To solve this optimization problem, [36] first **approximate this intractable optimization problem using Monte Carlo sampling via**

LP. Let $D = (x_1, y_1), ..., (x_n, y_n)$ be the uniformly sampled points at random from B, the approximation of the objective in Equation A.9 is

$$\min_{A_l, B_l, C_l} \sum_{i=1}^{n} (h(x_i, y_i) - (A_l \cdot x_i + B_l \cdot y_i + C_l))$$
$$\text{subject to } \bigwedge_{i=1}^{n} A_l \cdot x_i + B_l \cdot y_i + C_l \leq h(x_i, y_i). \tag{A.10}$$

This provides potentially unsound bounds, meaning that there can be points in region B that violate the bounds. To provide soundness, one can **adjust the offset to guarantee soundness utilizing Fermat's theorem**. Next, compute $\Delta_l = \min_{(x,y)\in B} h(x, y) - (A_l \cdot x + B_l \cdot y + C_l)$ and adjust the lower bound by updating the offset: $C_l \leftarrow C_l + \Delta_l$. To compute Δ_l, let $A_l \cdot x + B_l \cdot y + C_l$ be the initial lower bound in B obtained from the LP approximation.
For $h(x, y) = \sigma(x)\tanh(y)$, consider the extreme points of $F(x, y) = \sigma(x)\tanh(y) - (A_l \cdot x + B_l \cdot y + C_l)$ via its partial derivatives:

$$\frac{\partial F}{\partial x} = \sigma(x)\tanh(y)(1 - \sigma(x)) - A_l \tag{A.11}$$
$$\frac{\partial F}{\partial y} = \sigma(x)(1 - \tanh^2(y)) - B_l \tag{A.12}$$

Consider three different cases:
1. If $x \in \{l_x, u_x\}$ and $y \in [l_y, u_y]$, $\frac{\partial F}{\partial y} = 0$ can be written as

$$(1 - \tanh^2(y)) = B_l / S_x \tag{A.13}$$

where $S_x = \sigma(x)$ is a constant.

2. If $x \in [l_x, u_x]$ and $y \in \{l_y, u_y\}$, setting $\frac{\partial F}{\partial x} = 0$ becomes

$$\sigma(x)(1 - \sigma(x) = A_l / T_y \tag{A.14}$$

where $T_y = \tanh(y)$ is a constant.

3. Otherwise, consider both $\frac{\partial F}{\partial x} = 0$ and $\frac{\partial F}{\partial y} = 0$ to reduce $\tanh(y)$ and obtain

$$\sigma(x)^4 + (-2 - B_l)\sigma(x)^3 + (1 + 2B_l)\sigma(x)^2 + (-B_l)\sigma(x) - A_l^2 = 0. \tag{A.15}$$

According to Fermat's theorem on stationary points, $F(x, y)$ achieves its extremum at B either in the roots of Equation A.13, A.14 and A.15 or at the four corners of B. Thus, one can obtain $\Delta_l = \min_{(x,y)\in B} F(x, y)$ by evaluating F at these points and selecting the minimum among them.

For $h(x, y) = \sigma(x)y$, the analysis is similar and can be found in [36].

A.5 Normalization Layer

Normalize each vector v_j by subtracting the mean of its entries, i.e.

$$v_{j+1}^i = v_j^i - \text{mean}(v_j) = v_j - \frac{1}{N_j}\sum_{i=1}^{N_j} v_j^i$$

where N_j is the dimension of v_j for some $j > 1$, and v_j^i is its i-th entry. Note that the normalization layer is a linear layer, in that the normalized vector v_{j+1} can be expressed as

$$v_{j+1} = v_j - \frac{1}{N_j}\mathbf{1}v_j = \left(I - \frac{1}{N_j}\mathbf{1}\right) v_j$$

where $\mathbf{1}$ denotes the $N_j \times N_j$ matrix of all ones and I is the $N_j \times N_j$ identity matrix. Hence the associated abstract transformer is exact, i.e. the polyhedral upper and lower bounds agree and are given by the above expression.

A.6 Tanh

Following [43], let l_j and u_j be the concrete lower (resp. upper) bounds in the previous layer. Then we set $l_i' = \tanh(l_j)$ and $u_i' = \tanh(u_j)$. If $l_j = u_j$, then $a_i'^{\leq}(x) = a_i'^{\geq}(x) = \tanh(l_j)$. Otherwise, consider $a_i'^{\leq}(x)$ and $a_i'^{\geq}(x)$ separately. Let

$$\lambda = \frac{\tanh(u_j) - \tanh(l_j)}{u_j - l_j}$$

and

$$\lambda' = \min\left\{\tanh'(l_j), \tanh'(u_j)\right\}$$

If $0 < l_j$, then $a_i'^{\leq}(x) = \tanh(l_j) + \lambda(x_j - l_j)$, otherwise $a_i'^{\leq}(x) = \tanh(l_j) + \lambda'(x_j - l_j)$. If $u_j \leq 0$, then $a_i'^{\geq}(x) = \tanh(u_j) + \lambda(x_j - u_j)$, and $a_i'^{\geq}(x) = \tanh(u_j) + \lambda'(x_j - u_j)$ otherwise.

A.7 Polyhedral Verification for the dot product and Softmax

We considered three possible ways to bound the softmax function presented in the literature [42, 36, 50], of which we adopt the third approach as its bounds are provably tighter than the bounds derived from the other two methods [50].

A.7.1 Bounding multiplication, division, and exponential separately

[42] derive polyhedral bounds for multiplication, division, and for the exponential function. The desired bounds follow using function composition and linear transformations described below.

- **Multiplication:** Let x and y be scalars with upper and lower bounds u_x, u_y and l_x, l_y. Then

$$l_y x + l_x y - l_x l_y \leq xy \leq u_y x + l_x y - l_x u_y$$

- **Division:** Using the same setting as above, decompose the division operation as $x/y = x \cdot 1/y$, so that it suffices to derive bounds for the reciprocal function. Assume that $0 < l_y \leq y \leq u_y$ and denote $1/y =: \sigma(y)$. Then

$$\sigma'\left(\frac{u_y + l_y}{2}\right)\left[y - \left(\frac{u_y + l_y}{2}\right)\right] + \sigma\left(\frac{u_y + l_y}{2}\right) \leq \sigma(y) \leq \frac{\sigma(u_y) - \sigma(l_y)}{u_y - l_y}(y - l_y) + \sigma(l_y)$$

- **Exponential:** Proceeding as above,

$$\exp(d)(x - d) + \exp(d) \leq \exp(x) \leq \frac{\exp(u_x) - \exp(l_x)}{u_x - l_x}(x - l_x) + \exp(l_x)$$

  In the lower bound, let $d := \min((l_x + u_x)/2, l + 1 - \Delta_d)$, where Δ_d is a small positive number, such as 10^{-2}.

A.7.2 Polyhedral Verification for Division using Fermat

We can improve the bounds for division using Fermat's theorem as in [36]. For the lower bound, we assume that $l_x \leq x \leq u_x$ and $0 < l_x \leq y \leq u_y$ and define

$$F^L(x, y) = \frac{x}{y} - \left(\alpha^L x + \beta^L y + \gamma^L\right)$$

where α, β and γ are real numbers. The partial derivatives are given by

$$\frac{\delta F^L}{\delta x} = \frac{1}{y} - \alpha^L, \; \frac{\delta F^L}{\delta y} = -\frac{x}{y^2} - \beta^L$$

We distinguish between the following cases:

1. If $x \in \{l_x, u_x\}$, then $\frac{\delta F^L}{\delta y} = 0$ implies that $y^2 = -\frac{x}{\beta^L}$ (recall that $y > 0$ by assumption as in the previous subsection).

2. If $y \in \{l_y, u_y\}$, then $\frac{\delta F^L}{\delta y}$ is constant, so F is monotonous on the two boundaries, hence it suffices to consider the corner points.

We also note that there are no (isolated) minima inside the boundaries, i.e. within $(l_x, u_x) \times (l_y, u_y)$, because, for that to be the case, F^L would have to be positive definite at some point. However,

$$\frac{\delta^2 F^L}{\delta x^2} = 0, \; \frac{\delta^2 F^L}{\delta y^2} = \frac{2x}{y^3}, \; \frac{\delta^2 F^L}{\delta x \delta y} = -\frac{1}{y^2}$$

Hence for the determinant of the Hessian of F^L, we have

$$\frac{\delta^2 F^L}{\delta x^2} \cdot \frac{\delta^2 F^L}{\delta y^2} - \left(\frac{\delta^2 F^L}{\delta x \delta y}\right) = -\frac{1}{y^4} < 0$$

so that the eigenvalues of the Hessian cannot have the same sign, hence F^L cannot be positive definite.

In conclusion, to obtain the offset needed for a sound lower bound, it suffices to ensure that $F^L \geq 0$ at the corners of $[l_x, u_x] \times [l_y, u_y]$ and at the root (w.r.t. y and with $x = \{l_x, u_x\}$) of $y^2 = -\frac{x}{\beta^L}$, if it exists and lies within the region.

Computing the Lower Bound: Proceeding by analogy with [36], we first find an approximate a lower bound by solving

$$\min_{\alpha^L, \beta^L, \gamma^L} \sum_{i=1}^{n} \left[\frac{x_i}{y_i} - \left(\alpha^L x_i + \beta^L y_i + \gamma^L\right)\right]$$

where the x_i and y_i are sampled from $B = [l_x, u_x] \times [l_y, u_y]$. We use the parameters α^L, β^L, and γ^L found in this way to define the function $F^L(x, y)$

above, and let $\Delta^L = \min_{(x,y)\in B} F^L(x,y)$, for which it suffices to consider the four corners and the points $(l_x, \sqrt{-l_x/\beta^L}), (u_x, \sqrt{-u_x/\beta^L})$, if they exist (in B). The resulting sound bound is then

$$\alpha^L x + \beta^L y + \gamma^L + \Delta^L$$

A.7.3 Improved Softmax Bounds

Finally, the provably tighter bounds for softmax are introduced in [50]. They define

$$p_j = \frac{1}{1 + \sum_{j' \neq j} \exp\left(x_{j'} - x_j\right)}$$

for $x \in \mathbb{R}^K$ and $j = 1, ..., K$. For ease of notation, they focus on $j = 1$. All other cases follow by symmetry. They derive the following lower and upper bounds on the softmax output p_1:

$$L^{\mathrm{LSE}}(x) = \frac{\exp(x_1)}{\overline{\mathrm{SE}}(x; l, u)},$$
$$U^{\mathrm{LSE}}(x) = \frac{\underline{p}_1 \log(\overline{p}_1) - \overline{p}_1 \log(\underline{p}_1) - (\overline{p}_1 - \underline{p}_1)\mathrm{LSE}(\tilde{x})}{\log(\overline{p}_1) - \log(\underline{p}_1)},$$

where

$$\overline{\mathrm{SE}}(x; l, u) = \sum_{j=1}^{K} \left(\frac{u_j - x_j}{u_j - l_j} \exp l_j + \frac{x_j - l_j}{u_j - l_j} \exp u_j \right),$$
$$\mathrm{LSE}(\tilde{x}) = \log \left(\sum_{j=1}^{K} \exp x_j \right),$$
$$\overline{p}_1 = \frac{1}{1 + \sum_{j \neq 1} \exp(\tilde{l}_j)},$$
$$\underline{p}_1 = \frac{1}{1 + \sum_{j \neq 1} \exp(\tilde{u}_j)}.$$

Here, $\tilde{x}_j = x_j - x_1$, $\tilde{u}_j = u_j - l_1$, and $\tilde{l}_j = l_j - u_1$. They linearize $L^{\mathrm{LSE}}(x)$ and $U^{\mathrm{LSE}}(x)$ by using tangent planes to these bounds. Recall that, for a function $f : \mathbb{R}^K \to \mathbb{R}$, a tangent plane at a point $c \in \mathbb{R}^K$ can be described as

$$\overline{f}_c(x) = \sum_{j=1}^{K} \left(\frac{\delta f(c)}{\delta x_j} (x_j - c_j) \right) + f(c)$$

or in the linearized form we use in our implementation,

$$\overline{f}_c(x) = \sum_{j=1}^{K} \frac{\delta f(c)}{\delta x_j} x_j - \sum_{j=1}^{K} \frac{\delta f(c)}{\delta x_j} c_j + f(c)$$

In our case,

$$\begin{aligned}
\frac{\delta L^{\mathrm{LSE}}(x)}{\delta x_1} &= L^{\mathrm{LSE}}(x) - \exp(x_1) \frac{1}{\overline{\mathrm{SE}}(x;l,u)^2} \left(\frac{\exp(u_1) - \exp(l_1)}{u_1 - l_1} \right), \\
\frac{\delta L^{\mathrm{LSE}}(x)}{\delta x_i} &= -\exp(x_1) \frac{1}{\overline{\mathrm{SE}}(x;l,u)^2} \left(\frac{\exp(u_i) - \exp(l_i)}{u_i - l_i} \right), i \neq 1, \\
\frac{\delta U^{\mathrm{LSE}}(x)}{\delta x_1} &= -\frac{\overline{p}_1 - \underline{p}_1}{\log(\overline{p_1}) - \log(\underline{p_1})} \left(\frac{\exp(x_1)}{\mathrm{SE}(x)} - 1 \right), \\
\frac{\delta U^{\mathrm{LSE}}(x)}{\delta x_i} &= -\frac{\overline{p}_1 - \underline{p}_1}{\log(\overline{p_1}) - \log(\underline{p_1})} \frac{\exp(x_i)}{\mathrm{SE}(x)}, i \neq 1,
\end{aligned}$$

where $\mathrm{SE}(x) = \sum_{j=1}^{K} \exp(x_j)$.

B Models and Training

In this section, we introduce the detailed model architectures, training process, datasets, and hardware specifications.

B.1 Model Architectures

We did not use lexicon for all models, and we convert all images to grayscale single-channel images scaled to 20×100. For the standard STR model pipelines, the image is input into the TPS transformation module described in 3, where the linear projection and the grid generator creates the grid for the transformation, and the grid sampler is the bilinear sampler that produces the rectified image. The rectified image is then put into the feature extractor, sequence modelling and decoder head depicted in 2, where 2a shows the detailed architecture of the CTC decoder model and 2b shows the detailed architecture of the attention decoder model. For the ViTSTR model, the architecture is given in 4. The input image is directly input into the model, where the patch embedding cuts the image into 5×5 pieces and is linearly projected to 128 hidden dimensions, before adding the positional encoding. We use 5 layers of the Transformer block for the results in the main

Layer Type	Configuration
Input	20 × 100 × 1
Conv + ReLU	6 × 6 × 32, s:2, p:0
Conv + ReLU	5 × 5 × 64, s:1, p:2
Max Pooling	1 × 2, s:2
Batch Norm	-
Conv + ReLU	3 × 3 × 128, s:2, p:0
Conv + ReLU	3 × 3 × 128, s:1, p:1
Conv + ReLU	3 × 3 × 128, s:1, p:0
Batch Norm	-
Reshape	#frames×128
LSTM	64 hidden dim
Linear Projection	#frames ×#classes
Softmax	-
CTC Decoder	-

(a) CTC decoder model architecture

Layer Type	Configuration
Input	20 × 100 × 1
Conv + ReLU	6 × 6 × 32, s:2, p:0
Conv + ReLU	5 × 5 × 64, s:1, p:2
Max Pooling	1 × 2, s:2
Batch Norm	-
Conv + ReLU	3 × 3 × 128, s:1, p:1
Max Pooling	1 × 2, s:2
Conv + ReLU	3 × 3 × 128, s:1, p:1
Conv + ReLU	2 × 2 × 128, s:1, p:0
Batch Norm	-
Map to Sequence	7 × 128
LSTM	64 hidden dim
Linear Projections	64 hidden dim
Softmax	-
Attention Mul	-
LSTM	64 hidden dim
Generator	#frames×#classes

(b) Attention decoder model architecture

Table 2: The model architecture of the feature extractor, sequence modelling and decoder head for the standard STR models. Here `s` and `p` stand for stride and padding size, respectively.

Layer Type	Configuration
Input	20 × 100 × 1
Conv + ReLU	6 × 6 × 32, s:2, p:0
Max Pooling	2 × 2, s:2
Conv + ReLU	5 × 5 × 64, s:1, p:2
Max Pooling	2 × 2, s:2
Conv + ReLU	3 × 3 × 128, s:1, p:1
Average Pooling	1 output dim
Linear Projection	128 hidden dim
Grid Generator	-
Grid Sampler	-

Table 3: Model architecture for TPS transformation

paper, followed by reshaping and linear projection to output the sequence of labels. For all models, the number of output frames varies from 11 to 21 depending on the model, whereas and the number of classes is 42, which includes the alphabet, numbers and some punctuation symbols.

B.2 Datasets

MJSynth (MJ) [14] is a synthetic dataset designed for STR, containing 8.9M word images. The word generation process is as follows: font rendering; border and shadow rendering; background coloring; composition; applying projective distortions; blending with real-world images; and finally adding noise. The **SynthText (ST)** [13] dataset is another synthetically generated

Layer Type	Configuration
Input	$20 \times 100 \times 1$
Patch Embedding	5×5
Linear Projection	128 hidden dim
Positional Encoding	-
Transformer Blocks:	$\times$ 5
Layer Norm	-
Multi-Head Attention	128 hidden dim
Residule Connection	-
Layer Norm	-
Fully Connected + ReLU	128 hidden dim
Fully Connected + ReLU	256 hidden dim
Fully Connected + ReLU	128 hidden dim
Residule Connection	-
Reshape	#frames$\times$128
Linear Projection	#frames$\times$#classes

Table 4: Model architecture for the ViTSTR

dataset and was originally designed for scene text detection. Nevertheless, it has also been used for STR by cropping word boxes from larger images. ST has 5.5M training data once the word boxes are cropped and filtered for non-alphanumeric characters. We use both MJ and ST for training (14.4M images in total).

For evaluation, we use 6 datasets. **IIIT5K-Words** [27] is the dataset crawled from Google image searches, which consists of 2,000 images for training and 3,000 images for evaluation. **ICDAR2013 (IC13)** [18] was created for the ICDAR 2013 Robust Reading competition. We use images from the born-digital images task, which consists of 3564 images for training and 1439 images for evaluation. **ICDAR2015 (IC15)** [17] was created for the ICDAR 2015 Robust Reading competition. We use images from the focused scene text task that are captured by Google Glasses while under the natural movement of the wearer. The benchmark contains 4,468 images for training and 2,077 images for evaluation. **Street View Text (SVT)** [47] contains outdoor street images collected from Google Street View, which contains noisy, blurry, and/or low-resolution images. It consists of 257 images for training and 647 images for evaluation. **SVT Perspective (SVTP)** [33] is

also collected from Google Street View, in which many images contain perspective projections to mimic non-frontal viewpoints. It contains 645 images for evaluation. **CUTE80 (CUTE)** [35] is collected from natural scenes, of which many are curved text images. It contains 288 cropped images for evaluation. We use the training images from all these datasets as validation data for training, and we evaluated the models by applying STR-Cert on the evaluation images from these datasets.

B.3 Training Configurations and Hyperparameters

Experiments are carried out on a Linux server (Ubuntu 18.04.2) with two Intel Xeon Gold 6252 CPUs and six NVIDIA GeForce RTX 2080 Ti GPUs. All our algorithms are implemented in Python, where we adopt PyTorch [32] for implementing the training and certification algorithms.

We mainly follow the training procedure of Baek *et al.* [4]. We use the AdamW [20] optimizer for training, with learning rate 0.001, betas=(0.9, 0.999) and weight decay=0.0001. The training batch is 512 and the total number of training iterations is 100k. PGD adversarial training [26] is adopted, where we perturb the image up to some *perturbation budget* with 10 gradient ascent steps to maximize the prediction loss of the preturbed image. This adversarially attacked image is then fed into the neural network and trained using the standard training loss. Gradient clipping is used at magnitude 5. We validate the model every 1000 training iterations on the union of the training sets of IC13, IC15, IIIT, and SVT, to select the model with the highest accuracy on this set.

C Additional Experiments and Discussions

In this section, we present additional experimental results for STR-Cert on STR models and provide a discussion.

C.1 Robustness Certification for Models with Different Depth

We provide additional certification results for the CTC decoder models, attention decoder models, and ViTSTR with different depths compared to those used in the main paper. For the CTC and attention decoder models, we provide certification results for models with 6 convolutional layers in the feature extractor. For ViTSTR, we provide certification results for two additional models with different numbers of layers (4 and 6) of the

Model	ViTSTR: 4 Transformer blocks				ViTSTR: 6 Transformer blocks			
Datasets	$\epsilon = .001$	$\epsilon = .003$	$\epsilon = .005$	$\epsilon = .01$	$\epsilon = .001$	$\epsilon = .003$	$\epsilon = .005$	$\epsilon = .01$
IIIT5K	97.5%	75.5%	57.0%	26.0%	92.5%	65.0%	45.0%	18.0%
IC13	98.5%	88.0%	74.5%	43.0%	95.5%	78.0%	59.5%	28.5%
IC15	95.5%	59.5%	41.5%	12.5%	91.5%	49.5%	34.5%	7.5%
SVT	97.5%	67.0%	60.5%	31.0%	92.5%	56.5%	45.5%	21.5%
SVTP	96.5%	62.0%	44.0%	22.0%	91.0%	50.0%	30.5%	13.5%
CUTE	97.0%	81.0%	59.5%	27.0%	92%	67.5%	45.0%	19.5%

Table 5: % certified in the first 200 correctly classified instances for the ViTSTR model with different layers of Transformer blocks.

Transformer block. In 5, we present the percentage certified under various perturbation budgets. It can be seen that a higher number of layers reduces the percentage certified due to the increased depth for certification, where the error compounds. However, similar scalability trends with respect to the perturbation budget can be observed for all model depths.

For the standard STR pipelines, we also experiment on models with different depths. The depth for the decoders is constant, and we found the depth of TPS transformation only marginally affects the accuracy and percentage certified. In addition, it is infeasible to certify models with more then 1 layer of LSTM, so we only study models with different numbers of convolutional layers in the feature extractor. In 6, the results for CTC and attention decoder models with 6 layers of CNNs in the feature extractor are presented. A slight drop in percentage certified is observed, where the attention decoder model suffers more since the abstraction error from the feature extractor is compounded by certifying the two LSTMs in the model.

C.2 Robustness Against Rotation

Adversarial robustness against rotations has been considered using DeepPoly [43], where they certified inputs against the usual pointwise perturbation plus rotation angle θ within some range. The interval domain for each pixel under all rotation angles is used to create the adversarial region for the certification. This produces extremely wide pixel input intervals and effectively loses most of the information of the original image for large rotation intervals, making the final bounds too imprecise. To address this, Singh *et al.* [43] proposed to refine the adversarial region by segmenting the

Model	CTC decoder: 6 layer CNNs			Attention decoder: 6 layer CNNs		
Datasets	$\epsilon = .001$	$\epsilon = .003$	$\epsilon = .005$	$\epsilon = .001$	$\epsilon = .003$	$\epsilon = .005$
IIIT5K	96.5%	73.5%	43.5%	87.0%	63.0%	33.0%
IC13	97.5%	86.0%	55.5%	91.5%	77.5%	48.5%
IC15	93.5%	53.5%	15.0%	87.5%	31.0%	8.5%
SVT	93.0%	65.0%	30.5%	84.0%	38.0%	16.5%
SVTP	92.5%	59.5%	31.5%	85.5%	53.5%	18.0%
CUTE	96.0%	81.5%	37.0%	90.5%	67.5%	28.5%

Table 6: % certified in the first 200 correctly classified instances for CTC decoder and attention decoder model with 6 convolutional layers.

rotation range into n partitions and compute the adversarial region induced by each segment, before further segmenting each adversarial region's interval domain into m parts and merging the certification results for all these $n * m$ adversarial regions. Although this method in theory allows for certification against rotation, it is very computationally intensive and sometimes impractical. Singh *et al.* [43] used $n, m \approx 300$ to certify a single MNIST image against rotation between -45 to 65 degrees. For our case, we fail to certify a single sample against rotation within a reasonable computation time, and further increasing the granularity of the refinement would mean that each sample could take weeks to certify.

During experiments, we observed that, as expected, the STN module for the standard STR models is very robust against rotation, and the patch embedding for ViTSTR should also provide a certain level of robustness against rotation. Unfortunately, the rotation certification approach of DeepPoly is insufficient to prove that STR models are robust against rotation?. Therefore, developing certification tools specifically for rotation, perhaps based on an abstraction domain other than polyhedra, is a very interesting direction for future work.