Robustness Verification for Attention Networks using Mixed Integer Programming

Hsuan-Cheng Liao¹³, Chih-Hong Cheng²⁺, Maximilian Kneissl¹, Alois Knoll³

Abstract—Attention networks such as transformers have been shown powerful in many applications ranging from natural language processing to object recognition. This paper further considers their robustness properties from both theoretical and empirical perspectives. Theoretically, we formulate a variant of attention networks containing linearized layer normalization and sparsemax activation, and reduce its robustness verification to a Mixed Integer Programming problem. Apart from a naive encoding, we derive tight intervals from admissible perturbation regions and examine several heuristics to speed up the verification process. More specifically, we find a novel bounding technique for sparsemax activation, which is also applicable to softmax activation in general neural networks. Empirically, we evaluate our proposed techniques with a case study on lane departure warning and demonstrate a performance gain of approximately an order of magnitude. Furthermore, although attention networks typically deliver higher accuracy than general neural networks, contrasting its robustness against a similar-sized multi-layer perceptron surprisingly shows that they are not necessarily more robust.

I. INTRODUCTION

Over the past decade, neural networks (NNs) have become the de facto backbone in many applications and are integrated in safety-critical systems such as self-driving cars. In particular, attention networks (ATNs) such as transformers have been commonly found to be the best-performing models in terms of accuracy in several domains including natural language processing and object recognition [1], [2]. This paper further investigates their robustness properties, comparing the robustness of ATNs and multi-layer perceptrons (MLPs) of roughly equivalent sizes. The motivation stems from recent studies unveiling robustness issues of general NNs, pinpointing a potential weakness of these learning-based models where they fail to uphold the correct prediction in the event of small input perturbations [3], [4]. For example, [4] discovers that with a slight value change at a single pixel, the underlying convolutional NN (CNN) classifies the image to a wrong label with relatively high confidence.

To tackle the issue above, remedies such as adversarial training and robustness verification are introduced. The former attempts to strengthen the network with data-augmentation-like techniques [3] and the latter quantifies the network’s limitation formally [5]. In this paper, we consider the latter to analyze and compare the robustness performance of different networks. Conceptually, we reduce network robustness verification to an optimization problem that finds the minimum adversarial perturbation altering the network’s prediction. Then, the network is said more robust around the original input if the optimal minimum adversarial perturbation is larger and vice versa. To solve the optimization problem, we employ Mixed Integer Programming (MIP) thanks to its capability to deliver exact verification solutions. However, a particular challenge is to handle the self-attention mechanism in ATNs, which contains exponential functions, dot products and division. Toward this challenge, we focus on a variant of ATNs using sparsemax activation [6] and construct precise encoding into Mixed Integer Quadratically Constrained Programming (MIQCP) problems, circumventing exponential functions and retaining quadratic ones. In addition, we opt for linearized layer normalization as [7] suggests, avoiding division over small values and potentially exploding variable intervals. Finally, we explore a variety of heuristics, including a novel bounding technique over sparsemax activation and a perturbation region partitioning algorithm, to accelerate the verification process. The novel bounding technique is also applicable to softmax activation in general NNs, thereby increasing its usefulness for the community.

We demonstrate our methodology with an industrial use case - lane departure warning, which is essentially a time-series classification and regression problem. More specifically, the NN has to predict the direction and time to the lane departure, with a certain duration of past driving information such as ego vehicle velocity and time to collision to adjacent vehicles. Such a lane departure warning system can be used for human driving assistance or runtime monitoring of automated vehicle control functions. With our empirical studies, we surprisingly learn that although ATNs typically deliver higher accuracy compared to ReLU-based MLPs, they are not necessarily more robust than their counterparts. Our results hence indicate that it is crucial to show thorough studies and guarantees before deploying an NN-based product into the market. In summary, our contributions include the following:

- To exactly verify robustness of ATNs using MIQCP;

¹Hsuan-Cheng Liao and Maximilian Kneissl are with DENSO AUTOMOTIVE Deutschland GmbH, 85386 Eching, Germany h.liao, m.kneissl@eu.denso.com
²Chih-Hong Cheng is with Fraunhofer Institute for Cognitive Systems IKS, 80686 Munich, Germany chih-hong.cheng@iks.fraunhofer.de
³Hsuan-Cheng Liao and Alois Knoll are with Department of Informatics, Technical University of Munich, 85748 Garching, Germany knoll@in.tum.de
⁺Key results are conducted during the author's service in DENSO AUTOMOTIVE Deutschland GmbH
*This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 956123 - FOCETA
To propose accelerating heuristics for ATN robustness verification;
To evaluate proposed techniques with an industrial use case and benchmark NN robustness performance.

The rest of the paper is organized in the following way. Section II browses the relevant literature with emphases on verification methods for general NNs and ATNs; Section III introduces the variant of ATN concerned in this paper. Section IV then details the problem formulation as well as our heuristics for robustness verification, whose effectiveness and efficiency are demonstrated in Section V. Lastly, Section VI concludes with a few final remarks.

II. RELATED WORK

This section provides an overview of related work, focusing on robustness verification for piece-wise linear NNS and ATNs.

A. Robustness Verification for Neural Networks

Following a common categorization [8], [9], we introduce two main branches of verification methods, namely complete and incomplete ones. To illustrate the difference, we first assume an adversarial polytope to be the exact set of NN outputs resulting from the norm-bounded set of perturbed inputs. To assert the robustness of the NN on these perturbed inputs, complete methods handle the adversarial polytope directly, attaining an adversarial example or a robustness certificate for each of the inputs when given sufficient processing time. These methods usually apply Mixed Integer Programming (MIP) [5], [10], [9] or Satisfiability Modulo Theory (SMT) [11], [12], which in turn utilizes Linear Programming (LP) or Satisfiability (SAT) solvers with accelerating techniques such as interval analysis [10], [11] and presolve algorithms [9] in a Branch-and-Bound (BnB) fashion [13]. By contrast, incomplete methods reason upon an outer-approximation of the adversarial polytope. Such reasoning typically results in faster verification time yet possibly some peculiar queries since an incomplete verifier can only certify the robustness of a portion of robust inputs. Common methods in this branch include duality [8], abstract interpretation [14] and Semi-Definite Programming (SDP) [13]. For more details, interested readers are referred to the survey paper [15].

B. Robustness Verification for Attention Networks

ATNs typically host more complex operations, rendering them more difficult for verification. From the literature, we are only aware of two lines of work [7], [16], both conducted with sentiment classification in natural language processing. In [7], the authors calculate linear intervals for all operations in a transformer network to find the lower bound of the softmax values of the ground-truth class and the most-probable-other-than-ground-truth class. Given an admissible perturbation region, if the computed lower bound is larger than zero, the model is guaranteed to be robust since the prediction remains unchanged. Holding a similar strategy, the most recent work [16] applies abstract interpretation and suggests several techniques such as noise signal reduction and softmax summation constraint to achieve a better speed and precision during verification.

Our work differs from these verification frameworks in two ways. First, we find minimum adversarial perturbations through precise MIP encoding and fasten the procedure with novel tactics, allowing us to compare the robustness with standard MLPs. Second, we study lane departure warning, which prompts us to formulate exact robustness properties in both classification and regression problems.

III. PRELIMINARIES

This section provides a brief description of the ATN under verification in this paper. For more elaborate illustration of general ATNs, readers are encouraged to see [1], [2].

As shown in Fig. 1, an ATN typically processes an array of embedded tokens with alternating blocks of multi-head self-attention (MSA) and multi-layer perception (MLP), which are both preceded by layer normalization (LN) and followed by residual connections1. Then, after another layer normalization and token-wise mean extraction, the network is appended with suitable affine heads for downstream predictions such as classification (CLS) and regression (REG). Mathematically, given an input $x \in \mathbb{R}^{N \times D}$, in which $N$ is the number of tokens and $D$ the dimension of features, we write:

$$z^0 = x$$
$$z^l = \text{MSA}(\text{LN}(z^{l-1})) + z^{l-1},$$
$$z^l = \text{MLP}(\text{LN}(z^l)) + z^l,$$
$$f^{\text{CLS}}(x) = \text{CLS}(z^L) \in \mathbb{R}^C,$$
$$f^{\text{REG}}(x) = \text{REG}(z^L) \in \mathbb{R},$$

in which $z^L$ is the token-wise mean of $\text{LN}(z^L), C$ is the number of predefined classes and $l = 1, \ldots, L$ is the layer index.

As introduced, we place a sparsemax activation in MSA to foster precise MIQCP encoding, resulting in the following definition:

$$[Q_h, K_h, V_h] = zW_h^{QKV},$$
$$A_h = \text{sparsemax} \left( Q_h K_h^\top / \sqrt{D_h} \right),$$
$$S_h(z) = A_h V_h,$$
$$\text{MSA}(z) = [S_h(z), \ldots, S_h(z)] W^{\text{MSA}},$$

where $z \in \mathbb{R}^{N \times D}$ is a general input matrix, $Q_h, K_h, V_h \in \mathbb{R}^{D_h}$ are the query, key and value matrices for the $h$-th self-attention head, $H$ is the number of self-attention heads, $D_h$ is the dimension of each head, $W_h^{QKV} \in \mathbb{R}^{D \times (D_h^3)}$ and $W^{\text{MSA}} \in \mathbb{R}^{(D_h H) \times D}$ are the trainable weights, and $h = 1, \ldots, H$ is the head index. Essentially, sparsemax projects an input vector $u \in \mathbb{R}^D$ onto the probability simplex $\Delta^{D-1}$, and could serve as a linear approximation to softmax [6]. Algorithm 1 and Fig. 2 provide the calculation steps for a closed-form solution.

1Placing LN before MSA and MLP is found to give better network performance than placing it after residual addition [17].
and the visualization of a 2D case. Now with sparsemax, we shall be able to precisely encode MSA as it contains only (piece-wise) linear and quadratic functions.

**Algorithm 1 Calculate sparsemax activation [6]**

1. **Input**: $\mathbf{u} \in \mathbb{R}^D$
2. Sort $\mathbf{u}$ into $\mathbf{\hat{u}}$, where $\mathbf{\hat{u}}_1 \geq \cdots \geq \mathbf{\hat{u}}_D$
3. Find $k(\mathbf{\hat{u}}) := \max \left\{ k \in [1, D] \mid 1 + k \mathbf{\hat{u}}_k > \sum_{j \leq k} \mathbf{\hat{u}}_j \right\}$
4. Define $\tau(\mathbf{\hat{u}}) = \left( \frac{\sum_{i \leq k} \mathbf{\hat{u}}_i}{k(\mathbf{\hat{u}})} \right)^{-1}$
5. **Output**: $p \in \Delta^{D-1}$, where $p_i = \max (u_i - \tau(\mathbf{\hat{u}}), 0)$

**Fig. 2:** softmax vs. sparsemax, given an input vector $\mathbf{u} = [t, 0] \in \mathbb{R}^2$ [6].

To proceed with the ATN architecture, MLP is a two-layer NN with ReLU activation, written as $\text{MLP}(\mathbf{z}) = \text{ReLU}(\mathbf{z}W_{\text{MLP}}^1 + \mathbf{b}_{\text{MLP}}^1)W_{\text{MLP}}^2 + \mathbf{b}_{\text{MLP}}^2 \in \mathbb{R}^{N \times D}$, where $W_{\text{MLP}}^1 \in \mathbb{R}^{D \times D_{\text{MLP}}}$, $\mathbf{b}_{\text{MLP}}^1 \in \mathbb{R}^{D_{\text{MLP}}}$, $W_{\text{MLP}}^2 \in \mathbb{R}^{D_{\text{MLP}} \times D}$, $\mathbf{b}_{\text{MLP}}^2 \in \mathbb{R}^D$ are the trainable parameters. Likewise, the linearized LN with element-wise affine transformation is defined as $\text{LN}(\mathbf{z}) = \left[ w \odot (\mathbf{z}_1 - \mu_{\mathbf{z}_1}) + \mathbf{b} ; \ldots ; w \odot (\mathbf{z}_N - \mu_{\mathbf{z}_N}) + \mathbf{b} \right] \in \mathbb{R}^{N \times D}$, where $\odot$ denotes element-wise multiplication, $\mathbf{z}_1, \ldots, \mathbf{z}_N$ are the row vectors of $\mathbf{z}$, $\mu_{\mathbf{z}_1}, \ldots, \mu_{\mathbf{z}_N} \in \mathbb{R}$ are the row vector means, and $\mathbf{w} \in \mathbb{R}^D$ and $\mathbf{b} \in \mathbb{R}^D$ are the trainable parameters [18]. Such modification circumvents the division over relatively small input variance $\sigma_{\mathbf{z}}$ in the quadratic LN definition, thereby allowing the variables to be bounded more tightly.

Finally, we conduct an empirical study on our use case (see Appendix A) and show that the formulated ATN delivers similar accuracy performance to general ones that contain softmax activation and quadratic LN.

**IV. METHODOLOGY**

Having denoted the network operations, in this section, we first define robustness properties of classification and regression models. Then, we highlight our MIQCP encoding strategy and present several heuristics that shall accelerate the solving of the encoded MIQCP.

**A. Problem Formulation**

We formalize the problem of robustness verification as follows: Let $f(\cdot) : \mathbb{R}^M \rightarrow \mathbb{R}^N$ denote the network under verification, $\mathbf{x} \in \mathbb{R}^M$ the original data point on which the network is being verified, $\mathbf{g}(\mathbf{x}) \in \mathbb{R}^N$ the ground truth and...
$x' \in \mathbb{R}^M$ a perturbed input which tries to deceive the network, we write:

$$\min_{x'} \mathcal{D}_p(x', x)$$

subject to $x' \in \mathcal{R}_p(x)$,

$$f(x) = gt(x) \land f(x') \neq gt(x),$$

where $\mathcal{D}_p(\cdot, \cdot)$ is the $l_p$-distance with commonly used $p \in \{1, 2, \infty\}$ and $\mathcal{R}_p(x) = \{x' \mid \|x' - x\|_p \leq \varepsilon\}$ is the $l_p$-norm ball of radius $\varepsilon$ around $x$. Now, for a classification model, (12) can be extended into:

$$\arg\max_i (f_i^{\text{CLS}}(x)) = gt^{\text{CLS}}(x) \land \arg\max_i (f_i^{\text{CLS}}(x')) \neq gt^{\text{CLS}}(x),$$

where $gt^{\text{CLS}}(x) \in \{1, \ldots, C\}$ is the ground-truth class label and $f_i^{\text{CLS}}$ denotes the $i$-th element of the classification head output. As for a regression model, assuming a real-valued output. As for a regression model, (12) can be extended into:

$$\max (f_i^{\text{REG}}(x)) = gt^{\text{REG}}(x) \land \max (f_i^{\text{REG}}(x')) \neq gt^{\text{REG}}(x),$$

where $\zeta$ is a predefined tolerance threshold. Moreover, inspired by the related work on abstract interpretation [16], we note a set-based formulation that involves two perturbed variables, rewriting the entire optimization problem as:

$$\min_{x', x''} \mathcal{D}_p(x', x'')$$

subject to $x', x'' \in \mathcal{R}_p(x)$,

$$f^{\text{REG}}(x) - gt^{\text{REG}}(x) < \zeta \land \notag$$

$$f^{\text{REG}}(x') - gt^{\text{REG}}(x') > \zeta \land \notag$$

As seen, the optimizer’s main task is to find within an admissible perturbation region a perturbed data point that is closest to the original one and fulfills the misprediction constraints.

### B. MIQCP Encoding

We employ Mixed Integer Quadratically Constrained Programming (MIQCP) to solve the above optimization problems. For implementation, we need to encode all network operations into a MIQCP problem layer by layer. Most of them, such as affine transformation, ReLU activation and max operation, are discussed by the related work [5], [10], [9]. Hence, we explain here only the additional operations introduced due to the self-attention mechanism with sparse-max activation.

The most challenging part lies in sorting an input vector and finding its support as stated in Algorithm 1 Lines 2 and 3. For the former, we introduce a binary permutation matrix $P \in \mathbb{R}^{D \times D}$ that serves as an intermediate variable and encode the following equations into MIP constraints:

$$\sum_{i=1}^{D} P_{ij} = 1, \quad \text{for } j = 1, \ldots, D; \quad (18)$$

$$\sum_{j=1}^{D} P_{ij} = 1, \quad \text{for } i = 1, \ldots, D; \quad (19)$$

For the latter, we first define a vector $\rho \in \mathbb{R}^D$, where $\rho_k = 1 + k\delta_k - \sum_{j=k}^{D} \delta_j$, for $k = 1, \ldots, D$ (see Algorithm 1 Line 3) and introduce another intermediate binary variable $x' \in \mathbb{R}^D$ such that its entry is one if the corresponding $\rho$ element is positive. Then, we nominate a big-M notation with $M \in \mathbb{R}$ and write:

$$\rho_k \leq M \times x', \quad \text{for } k = 1, \ldots, D; \quad (22)$$

$$-\rho_k \leq M \times (1 - x'), \quad \text{for } k = 1, \ldots, D; \quad (23)$$

$$k(\delta) = \sum_{k=1}^{D} x'$$

As such, we attain a naïve MIQCP encoding for quantifying the ATN’s robustness performance.

### C. Acceleration Heuristics

As indicated by the prior art, one usually needs several acceleration heuristics to solve an encoded MIP problem efficiently, even when equipped with a strong solver. We present our considerations in the following.

1) Interval Analysis: Interval analysis, or interval bound propagation, has been widely studied and proven effective to aid MIP solving [9]. The central idea is that with sufficiently tight intervals propagated across the network, non-linear functions such as ReLU or max can be constrained to certain behaviors. For instance, if an input variable of ReLU is bounded by $[1, 2, 9, 6, 12]$, the function is certainly active and the output variable is bounded by the same interval. Notably, such intervals merely serve as boundary constraints for MIP variables, and do not replace the precise MIP encoding equations.

For our network under verification, we start from deriving the interval for the perturbed input variable $x'$ from the admissible $\ell_p$-ball with the radius $\varepsilon$, using $p = 1$ as an example and leaving $p = 2, \infty$ to Appendix B.

$$x' \in \mathcal{B}_1(x)$$

$$\Rightarrow 0 \leq \|x' - x\|_1 = \|\delta\|_1 \leq \varepsilon$$

$$\Rightarrow 0 \leq \sum_{d=1}^{D} |\delta_d| \leq \varepsilon$$

$$\Rightarrow 0 \leq |\delta_d| \leq \varepsilon$$

$$\Rightarrow -\varepsilon \leq \delta_d \leq \varepsilon$$

$$\Rightarrow x_d' \in \left[\frac{x_d - \varepsilon}{\delta_d + \varepsilon}\right] = [x_d - \varepsilon, x_d + \varepsilon],$$

where $d = 1, \ldots, D$ denotes the depth dimension. Subsequently across the network, there are two main types of
bounding techniques. The first one is summarized from common matrix-matrix multiplications, and is applicable to linearized layer normalization, affine transformation and dot operation. For instance, using underscores for lower bounds and overscores for upper bounds, we derive element-wise bounds for the perturbed dot product $QK^T$ (see (7)) as follows:

$$QK^T_{ij} = q_k k^T_j \in \left[ q_k^T, q_k^T \right]$$

$$= \left\{ \sum_{d=1}^{D} q_d' k_{jd}^T \sum_{d=1}^{D} q_d' k_{jd}^T \right\} \subseteq \left\{ \sum_{d=1}^{D} \min(q_d' k_{jd}^T, q_d' k_{jd}^T, q_d' k_{jd}^T, q_d' k_{jd}^T) \right\}$$

$$= \left\{ \sum_{d=1}^{D} \max(q_d' k_{jd}^T, q_d' k_{jd}^T, q_d' k_{jd}^T, q_d' k_{jd}^T) \right\}$$

for the token indices $i, j = 1, \ldots, N$. The second technique, which has not been seen in literature, is designed specifically for the perturbed sparsemax output $\mathbf{A}'$ (see (7)). Conceptually, we observe that the lower (upper) bound of an output element can be calculated by applying sparsemax to a vector consisting of this element’s input lower (upper) bound and other elements’ input upper (lower) bounds. Hence, with $\sigma(\cdot)$ denoting sparsemax, we write:

$$\mathbf{A}'_{ij} \in \left[ \mathbf{A}'_{ij}, \mathbf{A}'_{ij} \right] = \left[ \sigma(q_k k^T, \ldots, q_k k^T), \sigma(q_k k^T, \ldots, q_k k^T) \right]$$

for the token indices $i, j = 1, \ldots, N$. In our evaluation, we generally see much tighter intervals (e.g., $[0,0.1]$) than the simple sparsemax interval $[0,1]$ with this technique. Moreover, it is also applicable to softmax in general NNs, offering even better usefulness.

2) Region Partitioning: Region partitioning shares a similar idea to interval analysis, attempting to tighten variable intervals and obtain a solution faster. In particular, we divide the admissible $\epsilon$-ball into disjoint sub-regions and apply a divide-and-conquer strategy. For example, given a partition step $0 < \epsilon_{\text{step}} \leq \epsilon$ (see (26)), we first set the current sub-region lower bound $\epsilon_{\text{min}} = 0$ and the current sub-region upper bound $\epsilon_{\text{max}} = \epsilon_{\text{min}} + \epsilon_{\text{step}}$, and then run the verification process for this sub-region. If the verifier cannot find a solution in the current sub-region, we move on to the next one, by setting $\epsilon_{\text{min}} + = \epsilon_{\text{step}}$ and $\epsilon_{\text{max}} + = \epsilon_{\text{step}}$, until the entire admissible region is covered. As such, we generally obtain a tighter interval for the perturbed input variable:

$$0 \leq \epsilon_{\text{min}} \leq \|\mathbf{x}' - \mathbf{x}\|_1 = \|\delta\|_1 \leq \epsilon_{\text{max}} \leq \epsilon$$

$$\implies \epsilon_{\text{min}} \leq \sum_{d=1}^{D} |\delta_d| \leq \epsilon_{\text{max}}$$

$$\implies 0 \leq |\delta_d| \leq \epsilon_{\text{max}}$$

$$\implies -\epsilon_{\text{max}} \leq \delta_d \leq \epsilon_{\text{max}}$$

$$\implies \mathbf{x}_d \in [x_d - \epsilon_{\text{max}}, x_d + \epsilon_{\text{max}}].$$

The tightness of the variable intervals across the network now depends highly on the value of $\epsilon_{\text{max}}$. If $\epsilon_{\text{max}}$ grows towards $\epsilon$, then the variable intervals shall fall back to the ones shown in (30). Nonetheless, considering adversarial examples often appear closely around the original data point, we conjecture that a small region would have already contained one of them, offering a high possibility for a quick solution.

We summarize the recursive procedure of verifying with region partitioning in Algorithm 2, taking the problem formulation in (10)--(12) as an example. We use Gurobi 9.5 [19] as the solver for the encoded MIQCP $\mathcal{M}$ and denote in Line 3 the returned interim results from the solver, namely optimization status $s$ (optimal, timeout or infeasible), solution count $n$, objective $ob$, counterexample $x'$, solving

```
Algorithm 2 Verify with region partitioning and a time limit

1: procedure VERIFYMODEL($f, x, gt(x), \epsilon, \epsilon_{\text{step}}, \epsilon_{\text{min}}, \epsilon_{\text{max}}, t_{\text{limit}}$
2: Encode $f, x, gt(x)$ in MIQCP $\mathcal{M}$ as formulated in (10)-(12) for the sub-region of $\epsilon_{\text{min}}$ and $\epsilon_{\text{max}}$
3: Solve $\mathcal{M}$ with Gurobi under $t_{\text{limit}}$ and obtain interim results $s, n, ob, x', gap, t_{\text{exec}}$
4: if $s$ is optimal then
5: return (OPT, $ob, x'$)
6: else if $s$ is timeout and $n > 0$ then
7: return (SAT, $ob, x', \alpha$)
8: else if $s$ is timeout and $n = 0$ then
9: return (UNSAT, $\epsilon_{\text{min}}$
10: else if $s$ is infeasible then
11: $t_{\text{limit}} := t_{\text{exec}}$
12: $\epsilon_{\text{min}} += \epsilon_{\text{step}}$
13: $\epsilon_{\text{max}} = \min(\epsilon_{\text{max}} + \epsilon_{\text{step}}, \epsilon)$
14: if $\epsilon_{\text{min}} > \epsilon$ then
15: return (UNSAT)
16: else
17: return VERIFYMODEL($f, x, gt(x), \epsilon, \epsilon_{\text{step}}, \epsilon_{\text{min}}, \epsilon_{\text{max}}, t_{\text{limit}}$
18: end if
19: end if
20: end procedure
21: Input: $f, x, gt(x), \epsilon, \epsilon_{\text{step}}, t_{\text{limit}}$
22: Initialize: $\epsilon_{\text{min}} := 0, \epsilon_{\text{max}} := \epsilon_{\text{step}}$
23: result = VERIFYMODEL($f, x, gt(x), \epsilon, \epsilon_{\text{step}}, \epsilon_{\text{min}}, \epsilon_{\text{max}}, t_{\text{limit}}$
24: Output: The resulting tuple result
```
gap $\alpha$ and execution time $t_{\text{exec}}$. Additionally, to prevent the verifier from executing unboundedly, we place a time limit $t_{\text{limit}}$ on the overall algorithm. Lastly, apart from fastening the verification process, another advantage of implementing region partitioning in such a fashion is that one could attain a good lower bound of the optimal objective even if a timeout occurs. To illustrate, in Algorithm 2 Lines 8 and 9, the solver at least certifies that the model is robust within the $\epsilon_{\text{min}}$-ball even if it fails to find a solution within the time limit.

3) Variable Hints and Branching Priorities: We also consider passing variable hints and branching priorities to the MIQCP solver for faster optimization. Regarding the former, it has been reported that providing the solver with high-quality variable value hints shall guide it towards high-quality solutions better [19]. We examine two candidates: The first is the original data point $x$ itself, given that the perturbed data point $x'$ should be in its vicinity by definition, whereas the second is an adversarial data point $x_{\text{adv}}$, if exists, produced by the well-known Projected Gradient Decent attack [20] since the adversarial data point is already a feasible solution to the MIQCP problem. Subsequently, we propagate the value hint for the input variable over the network and obtain layer-by-layer hints accordingly.

We also hold two candidates for the latter heuristic, namely forward branching or backward branching. By intuition, given a neural network, the encoded integer variables located at earlier layers should affect those at later layers, thereby requiring higher priorities to be explored and computed. Nonetheless, considering adversarial attacks that essentially operate backpropagation of loss gradients, it might be reasonable to branch backward in the MIQCP instance. Hence, we implement both branching directions and conduct experiments to evaluate their strength.

V. EXPERIMENTAL RESULTS AND DISCUSSIONS

This section presents the experiment results of implementing the proposed techniques on verifying a network for lane departure warning. In the following, we first introduce the lane departure warning task and then conduct several ablation studies and NN robustness comparisons.

A. Lane Departure Warning

To date, most of the NN verification works consider image classification, sentiment analysis, or simple control tasks thanks to the low dimensionality. Although they all contribute substantially to verification methods and tools, some are criticized for their applicability to real-world systems. Therefore, inspired by [21], we regard a more industry-oriented use case, lane departure warning, which is fundamentally a time-series joint classification and regression task.

We utilize the HighD dataset\footnote{The utilization of the HighD dataset in this paper is for knowledge dissemination and scientific publication and is not for commercial use.} [22], a drone-recorded birds-eye-view highway driving dataset. More specifically, for each vehicle in the recordings, we process raw data into trajectory information, including the past 10 steps of time-wise features $x \in \mathbb{R}^{10 \times 14}$ (see Fig. 3 for the 14 features),

the direction $g_t^{\text{CLS}} \in \{0, 1, 2\}$ of and the time $g_t^{\text{REG}} \in [0, 1]$ to the actual lane departure. As mentioned, we present the training results of various NN architectures in Appendix A.

B. Ablation Studies

We now conduct ablation studies on the acceleration heuristics described in Section IV-C. Referring to Section III, the network under verification has the following hyper-parameters: $L = 1, H = 2, D_H = 4, D_{\text{MLP}} = 16$, equivalent approximately to a piece-wise linear feed-forward NN with 1600 neurons. Notably, we only allow the final token of the input variable $x$ to be perturbed, resulting in an encoded MIQCP with roughly 4000 linear constraints, 700 quadratic constraints, 400 general constraints (e.g., max or absolute operations) and 2500 binary variables. We test the heuristics with five random samples from the curated dataset and summarize the results in Table I. All experiments are run with an Intel i9-10980XE CPU @ 3.0GHz using 18 threads and 20 GB of RAM\footnote{We leave the experimentation of verifying the regression branch for future work.}.

It is observed that interval analysis aids the solver indeed, to a more significant extent, if the novel sparsemax bounding technique is applied. Although variable hints and branching priorities do not help much, we see a subtle improvement in the performance if the solver receives a feasible adversarial data point and is set to branch forward in the same direction of NN data flow. Additionally, region partitioning further reduces verification time as expected. However, it might be required to identify the best epsilon step for different experiments as we discover its magnitude does not necessarily correlate to verification speed. Lastly, combining interval analysis and region partitioning gives approximately an order of magnitude performance gain.

C. NN Robustness Comparisons

Finally, we conduct experiments to compare the robustness of an ATN and an MLP of equivalent sizes. To construct the MLP, we replace MSA of the ATN with another MLP of hidden-layer dimension $D_{\text{MLP}} = 16$. Then, verifying the MLP follows similar steps as prescribed in Section IV except that the encoding is relatively simpler and that the encoded problem can be solved by Mixed Integer Linear Programming (MILP). We set $\varepsilon = 0.03$ and $p = 1$ for the admissible

![Fig. 3: List of 14 features at each time step for the ego vehicle. The time to collision (TTC) towards a specific adjacent car is calculated by dividing the relative distance with the relative velocity.](image-url)
The admissible perturbation region can be derived from input feature values analytically for better physical interpretability. For example, we can set \( \varepsilon \) as the normalized value of ego car lateral acceleration, considering it a decisive feature for lane departure warning.

\( \ell_p \)-ball\(^5\), and combine the best performing technique in each sub-group from the previous section for verification.

We collect results from 60 valid points (on which ATN and MLP predict correctly before perturbation) and plot the robustness certificates in Fig. 4. Since verifying the ATN on some data points still takes much time, we report the lower bounds of the robustness values for data points requiring more than one hour to verify. This means that the points marked by “lower” in Fig. 4 can be further pushed to the right if the verifier is given more time. However, we first observe that the data samples are quite equally distributed across the upper and lower triangles. In addition, there is a dense cluster lying in the upper left corner, indicating that the MLP delivers a much larger minimum adversarial perturbation than the ATN for a significant portion of the data. Accordingly, although ATNs commonly generate higher accuracy than MLPs, they are not necessarily more robust than their counterparts, as opposed to the findings suggested by some related work on vision tasks [23], [24]. Based on such outcomes, we infer that NNs generally perform differently in diverse domain tasks, and that it is necessary to conduct thorough studies and give guarantees on both NN accuracy and robustness properties before deploying NN-based products to the market.

### VI. Conclusion

This paper investigates the robustness properties of attention networks (ATNs) and multi-layer perceptrons (MLPs). We formulate a variant of ATNs, encode it into a Mixed Integer Quadratically Constrained Programming (MIQCP) problem and propose verification accelerating heuristics. Our proposals fasten the verification process for roughly an order of magnitude, compared to the naïve implementation. We conduct experiments with a lane departure warning system and discover that ATNs are not necessarily more robust than MLPs. Based on the results, we list out potential future directions. Firstly, it is necessary to keep exploring possibilities to verify larger and more complex networks efficiently for industrial applications. Secondly, similar to most related work, the paper only examines point-wise robustness properties of the networks. It is recommended that future verification work combine such analyses with systematic sampling and testing methods to give formal and statistical guarantees on safety-critical applications.

### APPENDIX

The following sections are only for reference, presenting supplementary materials to the dedicated sections above.

---

\(^5\)The admissible perturbation region can be derived from input feature values analytically for better physical interpretability. For example, we can set \( \varepsilon \) as the normalized value of ego car lateral acceleration, considering it a decisive feature for lane departure warning.
A. NN Training Results on Lane Departure Warning

We present in Table II the training results of various NN architectures. It can be seen that ATNs indeed deliver higher accuracy than their corresponding MLPs, and that our adaptations of sparsemax activation and linearized LN for the ATN do not degrade the performance much.

**Table II: Accuracy performance of various NN architectures. L denotes the number of layers in the network as defined in Section III. For layer normalization LN, 2 denotes the quadratic variant, 1 denotes the linear variant, and 0 means there is no layer normalization.**

| L | Network | Activation | LN | Accuracy | CLS REG |
|---|---------|------------|----|----------|---------|
| 1 | ATN     | sparsemax  | 2  | 98.91%   | 87.24%  |
|   |         | 1         | 98.31% | 88.46%   |          |
|   |         | 0         | 98.34% | 90.38%   |          |
| 2 | ATN     | sparsemax  | 2  | 98.01%   | 87.11%  |
|   |         | 1         | 97.95% | 87.56%   |          |
|   |         | 0         | 98.21% | 88.03%   |          |
|   | MLP     | –         | 2  | 97.62%   | 83.81%  |
|   |         | 1         | 97.46% | 83.55%   |          |
|   |         | 0         | 97.68% | 84.27%   |          |

B. Interval Analysis on the Input Variable

Based on Section IV-C, we complete the derivation of the interval for the perturbed input variable from the admissible \( \ell_p \)-ball with radius \( \varepsilon \). We first derive the interval for \( p = 2 \) as follows:

\[
\mathbf{x}' \in \mathbb{B}_2(\mathbf{x}) \quad \Rightarrow \quad \|\mathbf{x}' - \mathbf{x}\|_2 = \|\delta\|_2 \leq \varepsilon \tag{38}
\]

\[
\sqrt{\sum_{d=1}^{D} \delta_d^2} \leq \varepsilon \tag{39}
\]

\[
\sum_{d=1}^{D} \delta_d^2 \leq \varepsilon^2 \tag{40}
\]

\[
\delta_d^2 \leq \varepsilon^2 \quad \forall d \in [1, \ldots, D] \tag{41}
\]

\[
\delta_d \leq \varepsilon \quad \forall d \in [1, \ldots, D] \tag{42}
\]

\[
\mathbf{x}'_d \in \left[ \mathbf{x}_d - \varepsilon, \mathbf{x}_d + \varepsilon \right] \tag{43}
\]

where \( d = 1, \ldots, D \) denotes the depth index. Similarly, for \( p = \infty \), we write:

\[
\mathbf{x}' \in \mathbb{B}_\infty(\mathbf{x}) \quad \Rightarrow \quad \|\mathbf{x}' - \mathbf{x}\|_\infty = \|\delta\|_\infty \leq \varepsilon \tag{44}
\]

\[
\max_d |\delta_d| \leq \varepsilon \tag{45}
\]

\[
-\varepsilon \leq \delta_d \leq \varepsilon \tag{46}
\]

\[
\mathbf{x}'_d \in \left[ \mathbf{x}_d - \varepsilon, \mathbf{x}_d + \varepsilon \right] \quad \forall d \in [1, \ldots, D] \tag{47}
\]

Finally, if region partitioning is to be implemented (with the running \( \ell_p \)-ball radius bounds \( \varepsilon_{\text{max}} \) and \( \varepsilon_{\text{min}} \)), the derivations for \( p = 2, \infty \) follow a similar fashion and generate the same interval \( [\mathbf{x}_d - \varepsilon_{\text{max}}, \mathbf{x}_d + \varepsilon_{\text{max}}] \) as given in Section IV-C.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in NeurIPS, 2017.
[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” in ICLR, 2021.
[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” in ICLR, 2015.
[4] J. Su, D. V. Vargas, and K. Sakurai, “One pixel attack for fooling deep neural networks,” IEEE Trans. Evol. Comput., vol. 23, p. 828–841, 2019.
[5] A. Lomuscio and L. Maganti, “An approach to reachability analysis for feed-forward relu neural networks,” 2017.
[6] A. F. T. Martins and R. F. Astudillo, “From softmax to sparsemax: A sparse model of attention and multi-label classification,” in ICML, 2016.
[7] Z. Shi, H. Zhang, K.-W. Chang, M. Huang, and C.-J. Hsieh, “Robustness verification for transformers,” in ICLR, 2020.
[8] E. Wong and J. Z. Kolter, “Provably defenses against adversarial examples via the convex outer adversarial polytope,” in ICML, 2018.
[9] V. Tjeng, K. Xiao, and R. Tedrake, “Evaluating robustness of neural networks with mixed integer programming,” in ICLR, 2019.
[10] C.-H. Cheng, G. Nüñenberg, and H. Ruess, “Maximum resilience of artificial neural networks,” 2017.
[11] R. Ehlers, “Formal verification of piece-wise linear feed-forward neural networks,” in AVTA, 2017.
[12] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, “Reluplex: An efficient smt solver for verifying deep neural networks,” in CAV, 2017.
[13] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C.-J. Hsieh, and J. Z. Kolter, “Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification,” 2021.
[14] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, “AI2: Safety and robustness certification of neural networks with abstract interpretation,” in SP, 2018.
[15] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, “A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability,” Comput. Sci. Rev., vol. 37, p. 100270, 2020.
[16] G. Bonaert, D. I. Dimitrov, M. Baader, and M. Vechev, “Fast and precise certification of transformers,” in PLDI, 2021.
[17] R. Xiaoy, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T.-Y. Liu, “On layer normalization in the transformer architecture,” in ICLR, 2020.
[18] Y.-L. Hsieh, M. Cheng, D.-C. Juwu, W. Wei, W.-L. Hsu, and C.-J. Hsieh, “On the robustness of self-atteventive models,” in ACL, 2019.
[19] Gurobi Optimization, LLC, “Gurobi optimizer reference manual,” 2021.
[20] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in ICLR, 2018.
[21] V. Mahajan, C. Katrakazas, and C. Antoniou, “Prediction of lane-changing maneuvers with automatic labeling and deep learning,” TRR Journal, vol. 2674, pp. 336–347, 2020.
[22] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, “The highd dataset: A high dimensional real-world dataset for validation of highly automated driving systems,” in ITSC, 2018.
[23] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and M. Vechev, “AI2: Safety and robustness certification of neural networks with abstract interpretation,” in ICLR, 2018.
[24] R. Shao, Z. Shi, J. Yi, P.-Y. Chen, and C.-J. Hsieh, “On the adversarial robustness of vision transformers,” in UCCV, 2021.