Are Attention Networks More Robust? Towards Exact Robustness Verification for Attention Networks

Hsuan-Cheng Liao\textsuperscript{†,‡}, Chih-Hong Cheng\textsuperscript{§,++}, Hasan Esen\textsuperscript{†}, Alois Knoll\textsuperscript{†}

Abstract—As an emerging type of Neural Networks (NNs), Attention Networks (ATNs) such as Transformers have been shown effective, in terms of accuracy, in many applications. This paper further considers their robustness. More specifically, we are curious about their maximum resilience against local input perturbations compared to the more conventional Multi-Layer Perceptrons (MLPs). Thus, we formulate the verification task into an optimization problem, from which exact robustness values can be obtained. One major challenge, however, is the non-convexity and non-linearity of NNs. While the existing literature has handled the challenge to some extent with methods such as Branch-and-Bound, the additional level of difficulty introduced by the quadratic and exponential functions in the ATNs has not been tackled. Our work reduces this gap by focusing on sparsemax-based ATNs, encoding them into Mixed Integer Quadratically Constrained Programming problems, and proposing two powerful heuristics for a speedup of one order of magnitude. Finally, we train and evaluate several sparsemax-based ATNs and similar-sized ReLU-based MLPs for a lane departure warning task and show that the former is surprisingly less robust despite generally higher accuracy.

I. INTRODUCTION

Over the past decade, Neural Networks (NNs) have been widely adopted for many applications, including self-driving cars. Lately, as an emerging type of NNs, Attention Networks (ATNs) such as Transformers [1], are often found to be the most effective models in these applications, compared to the more conventional Multi-Layer Perceptrons (MLPs) [2]. Nevertheless, these studies primarily focus on accuracy performances. Parallel research has shown that NNs often lack robustness against input changes such as adversarial attacks and domain shifts [3], [4], hence hindering the dependability of the NN-based applications. Based on such background, a natural question is whether ATNs are also more robust than MLPs, given the better accuracy observed by many studies. To answer this question, we evaluate the maximum resilience of the ATNs against local input perturbations generally modeled by \( l_p \)-distances, i.e., exact robustness verification (where \( p \) can be 1, 2, ..., \( \infty \)). Ultimately, the goal is to hold a direct comparison of the robustness of the two kinds of NNs and gain insights from the results.

In the literature, research efforts have been made to enable exact robustness verification for MLPs, particularly ones with feed-forward layers and ReLU activation function [5], [6], [7], [8]. The main approach is to employ an optimization framework with which the optimal robustness performance can be computed based on constraints according to the network architecture and admissible input perturbations. However, there is still a gap for ATNs due to the more complex operations in the network, namely the dot product between variables and the activation function in the Multi-Head Self-Attention (MSA) block [1]. We attempt to close the gap in this work but should state that our proposal is merely a partial solution. To elaborate, having formulated the task into a Mixed Integer Programming (MIP)-based optimization problem as in [6], [7], we focus on the group of ATNs that uses sparsemax (instead of softmax) for MSA activation. This allows for a precise encoding of the network into the MIP, or more particularly Mixed Integer Quadratically Constrained Programming (MIQCP) due to the remaining quadratic terms. Despite a limited resolution (similar to the studies focusing only on ReLU-based MLPs), our work shows that sparsemax-based ATNs tend to be less robust than similar-sized MLPs despite generally higher accuracy. Consequently, our ultimate call for contrasting accuracy and robustness performances of NNs is served.

Additionally, in our work, we devise and apply two effective heuristics for accelerating the solving of the MIP problem by one order of magnitude. Notably, these heuristics are not restricted to our work but can be commonly implemented by related studies. Regarding the experiment, we consider the industrial application of Lane Departure Warning (LDW), which is essentially a time-series classification and regression problem. More specifically, the NN of interest 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 an LDW system can be used for human driving assistance or runtime monitoring of automated vehicle control functions. As mentioned, our empirical findings indicate that conducting thorough studies and providing rigorous guarantees is crucial before deploying any NN-based applications. In summary, our contributions include the following:

- To implement exact robustness verification for sparsemax-based ATNs;
- To propose accelerating heuristics for general robustness calculations.
verification;

- To benchmark ATN and MLP accuracy and robustness with an industrial application (i.e., LDW).

II. RELATED WORK

This section provides an overview of related work, focusing on robustness verification for ReLU-based MLPs (i.e., piece-wise linear feed-forward NNs) and ATNs, respectively.

A. Robustness Verification for Neural Networks

Following the common categorization [7], we introduce two main branches of verification methods: complete and incomplete. 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], [6], [7] or Satisfiability Modulo Theory (SMT) [8], [9], which in turn utilizes Linear Programming (LP) or Satisfiability (SAT) solvers with accelerating techniques such as interval analysis [6], [7], [8] or region partitioning [10] in a Branch-and-Bound (BnB) fashion [11]. By contrast, incomplete methods reason upon an outer approximation of the adversarial polytope. Such reasoning typically results in faster verification time yet possibly some robust queries being evaluated non-robust due to the over approximation. Common methods in this branch include duality [12], abstract interpretation [13] and Semi-Definite Programming (SDP) [11]. For more details, interested readers are referred to the survey paper [14].

B. Robustness Verification for Attention Networks

As ATNs contain more complex operations than other NNs, such as MLPs, they are typically more challenging to verify. From the literature, we are only aware of two lines of work [15], [16], both conducted with sentiment classification in natural language processing. In [15], the authors calculate linear intervals for all operations in a Transformer to find the lower bound of the difference between the softmax values of the ground-truth class and the most-probable-other-than-ground-truth class. Given an admissible perturbation region (i.e., input set), if the computed lower bound is larger than zero, the model is guaranteed robust since the region (i.e., input set), if the computed lower bound is larger than zero, the model is guaranteed 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 symbol reduction and softmax summation constraint to achieve better speed and precision during verification.

Our work differs from these verification frameworks in two ways. First, we find ATN’s maximum resilience (i.e., minimum adversarial perturbation) through precise MIP encoding and fasten the procedure with novel tactics, allowing us to compare the robustness with common MLPs. Second, we study LDW, an industrial application that has not been used for robustness verification.

III. PRELIMINARIES

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

Fig. 1: The Attention Network under verification [1], [2].

As shown in Fig. 1, an ATN typically processes an array of embedded tokens with alternating blocks of MSA and MLP, which are both preceded by Layer Normalization (LN) and followed by residual connections. 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:

$$z^0 = x$$  \hspace{1cm} (1)

$$\hat{z}^l = \text{MSA}(\text{LN}(z^{l-1}))+z^{l-1},$$  \hspace{1cm} (2)

$$z^l = \text{MLP}(\text{LN}(z^{l}))+\hat{z}^l,$$  \hspace{1cm} (3)

$$f^{\text{CLS}}(x) = \text{CLS}(\tilde{z}^L) \in \mathbb{R}^{C},$$  \hspace{1cm} (4)

$$f^{\text{REG}}(x) = \text{REG}(\tilde{z}^L) \in \mathbb{R}^{L},$$  \hspace{1cm} (5)

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

As introduced, we consider sparsemax-based MSA and provide its definition as follows:

$$[Q_h, K_h, V_h] = zW^{QKV},$$  \hspace{1cm} (6)

$$A_h = \text{sparsemax} \left( Q_h K_h^\top / \sqrt{D_H} \right),$$  \hspace{1cm} (7)

$$\text{SA}_h(z) = A_h V_h,$$  \hspace{1cm} (8)

$$\text{MSA}(z) = [\text{SA}_1(z), \ldots , \text{SA}_H(z)] W^{\text{MSA}},$$  \hspace{1cm} (9)

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.

3Placing LN before MSA and MLP is found to give better network performance than placing it after residual addition [17].
network is being verified, and \( \mathbf{x}' \in \mathbb{R}^M \) a perturbed input which tries to deceive the network, we write:

\[
\begin{align*}
\min_{\mathbf{x}'} \mathcal{D}_p(\mathbf{x}', \mathbf{x}) \\
\text{subject to } \mathbf{x}' \in \mathcal{B}_p(\mathbf{x}),
\end{align*}
\]

where \( \mathcal{D}_p(\cdot, \cdot) \) is the \( L_p \)-distance with commonly used \( p \in \{1, 2, \infty\} \) and \( \mathcal{B}_p(x) = \{ \mathbf{x}' | \| \mathbf{x}' - \mathbf{x} \|_p \leq \varepsilon \} \) the \( L_p \)-norm ball of radius \( \varepsilon \) around \( \mathbf{x} \), \( \text{gt}^{\text{CLS}}(\mathbf{x}) \in [1, \ldots, C] \) the ground-truth class label and \( f_i^{\text{CLS}} \) the \( i \)-th element of the classification head output. Conceptually, the optimizer’s main task is to find within an admissible perturbation region a perturbed data point closest to the original one and fulfills the misprediction constraints.

\section{B. MIQCP Encoding}

For verification, we need to encode all network operations into the formulated optimization problem (which will essentially be a MIQCP problem). Most of them, such as affine transformation and ReLU, are discussed by the related work [5], [6], [7]. Hence, we explain only the additional operations introduced by \text{sparsemax}, namely sorting (Algorithm 1, Line 2) and finding the support (Algorithm 1, Line 3).

For sorting, we introduce a binary integer permutation matrix \( \mathbf{P} \in \{0, 1\}^{D \times D} \) and encode the following constraints:

\[
\begin{align*}
\sum_{j=1}^{D} \mathbf{P}_{ij} & = 1, \quad \text{for } i = 1, \ldots, D; \\
\sum_{i=1}^{D} \mathbf{P}_{ij} & = 1, \quad \text{for } j = 1, \ldots, D;
\end{align*}
\]

\begin{equation}
\mathbf{u} = \mathbf{P} \mathbf{u}_t,
\end{equation}

\begin{equation}
\hat{\mathbf{u}}_i \geq \hat{\mathbf{u}}_{i+1}, \quad \text{for } i = 1, \ldots, D - 1,
\end{equation}

where \( \mathbf{u} \in \mathbb{R}^D \) is the (perturbed) input vector at \text{sparsemax} and \( \mathbf{u}_t \) the sorted output. For the support, we first define a vector \( \mathbf{p} \in \mathbb{R}^D \), where \( \rho_k = 1 + \delta \mathbf{u}_k - \sum_{j=1}^{k-1} \mathbf{u}_j \) \( (k = 1, \ldots, D) \), and then introduce another binary integer variable \( \zeta \in \{0, 1\}^D \) such that:

\begin{equation}
\zeta_k = \begin{cases} 
1, & \text{if } \rho_k > 0; \\
0, & \text{otherwise}.
\end{cases}
\end{equation}

For its actual computation, we implement the Big-M method [19] with large positive constants \( M^{\pm} \) and a small positive constant \( \eta \) (e.g., \( 10^{-6} \)) as follows:

\[
\begin{align*}
\rho_k & \leq M^+ \times \zeta_k, \\
-\rho_k + \eta & \leq M^- \times (1 - \zeta_k).
\end{align*}
\]

We defer the details for setting \( M^{\pm} \) in Lemma 1 and Lemma 2. For now, finding the support is equivalent to summing up the binary integer vector as \( s(\mathbf{u}) = \sum_{k=1}^{D} \zeta_k \). Lastly, Algorithm 1 Line 4 can be encoded with a linear constraint.

**Algorithm 1 Calculate \text{sparsemax} activation [18]**

1. **Input:** \( \mathbf{u} \in \mathbb{R}^D \)
2. Sort \( \mathbf{u} \) into \( \hat{\mathbf{u}} \), where \( \hat{\mathbf{u}}_1 \geq \cdots \geq \hat{\mathbf{u}}_D \)
3. Find \( s(\hat{\mathbf{u}}) := \max \left\{ k \in [1, D] | 1 + k \hat{u}_k > \sum_{j=1}^{k} \hat{u}_j \right\} \)
4. Define \( \tau(\hat{\mathbf{u}}) = \frac{(\sum_{i=1}^{D} \hat{u}_i) - 1}{s(\hat{u})} \)
5. **Output:** \( \mathbf{p} \in \mathbb{R}^D \), where \( \mathbf{p}_i = \max(\mathbf{u}_i - \tau(\hat{\mathbf{u}}), 0) \)
for the summation term and a quadratic constraint for the division. As such, we arrive at a plain MIQCP encoding for quantifying the sparsemax-based ATN’s robustness.

**Lemma 1.** For all $k = 1, \ldots, D$, the smallest value for the Big-M encoding in (19) is $\text{opt}_M^k = 1$.

**Proof.** We first rewrite $\rho_k = 1 + k \hat{u}_k - \sum_{j=1}^{\hat{k}} \hat{u}_j = 1 + \sum_{j=1}^{\hat{k}} (\hat{u}_k - \hat{u}_j)$. Now, with the sorting result (i.e., $\hat{u}_1 \geq \hat{u}_2 \geq \ldots \geq \hat{u}_{\hat{k}}$), it follows that $1 = \rho_1 \geq \rho_2 \geq \ldots \geq \rho_{\hat{k}}$, hence the lemma. □

**Lemma 2.** For all $k = 1, \ldots, D$, let the input of sparsemax be bounded as $u_k \in [u_{kL}, u_{kU}]$ and $\eta = 10^{-6}$. We first define $\lambda_k \in \mathbb{R}_+: \lambda_k = 1 + (k-1)(\hat{u} - \bar{u})$, where $\eta = \min(u_1, \ldots, u_D)$ and $\bar{u} = \max(u_1, \ldots, u_D)$. Then, the smallest value for the Big-M encoding in (20) is $\text{opt}_M^k = |\lambda_k| + \eta$, if $\lambda_k \leq 0$; otherwise, (20) needs not be implemented.

**Proof.** After sorting on $u$, we can only rely on vector-wise bounds for estimating $\hat{u}$, i.e., $\hat{u}_k \in [\hat{u}_kL, \hat{u}_kU]$. Considering the result of sorting (i.e., $\hat{u}_1 \geq \hat{u}_2 \geq \ldots \geq \hat{u}_{\hat{k}}$), we can then derive $\rho_k = 1 + k \hat{u}_k - \sum_{j=1}^{\hat{k}} \hat{u}_j = 1 + \sum_{j=1}^{\hat{k}} (\hat{u}_k - \hat{u}_j) \geq 1 + (k-1)(\hat{u} - \bar{u}) = \lambda_k$, where, there are two cases: If $\lambda_k \leq 0$, then the smallest value for (20) is $\text{opt}_M^k = |\lambda_k| + \eta$. Otherwise, $\rho_k \geq \lambda_k > 0$ and (20) is not needed. □

**C. Acceleration Heuristics**

As the prior art indicates, one usually needs several acceleration heuristics to solve an encoded MIP problem efficiently. We present our proposals in the following.

1) Interval Analysis: Interval analysis has been widely studied and proven effective in aiding MIP solving [6], [7]. The central idea is that with tight interval bounds propagated across the network, non-linear functions such as ReLU or max and (20) is not needed.

2) Progressive Verification with Norm-Space Partitioning: Space partitioning shares a similar goal to interval analysis, attempting to tighten variable bounds and generate a faster solution. The prior art has focused on how to do partitioning and prioritizing [10]. Our work differs slightly by observing that adversarial examples usually appear close to the clean input and exploiting the nature of the formulated optimization problem. We propose a progressive verification procedure based on norm-space partitioning. To illustrate, we divide the admissible $\ell_p$-ball into disjoint sub-regions and apply a divide-and-conquer strategy. For example, given a partition step $0 < \epsilon_{\text{step}} \leq \epsilon$, we first set the current sub-region lower bound $\epsilon_{\min} = 0$ and the current sub-region upper bound $\epsilon_{\max} = \epsilon_{\min} + \epsilon_{\text{step}}$, 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 $\epsilon_{\min} = +\epsilon_{\text{step}}$ and $\epsilon_{\max} = +\epsilon_{\text{step}}$, until the entire admissible region is covered. As such, we generally obtain a tighter interval for the perturbed input variable (taking $p = 1$, for instance):

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

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

$$\Rightarrow 0 \leq |\delta_d| \leq \epsilon_{\max}$$

$$\Rightarrow -\epsilon_{\max} \leq \delta_d \leq \epsilon_{\max}$$

$$\Rightarrow x_d \in [x_{dL}, x_{dU}] = [x_d - \epsilon_{\max}, x_d + \epsilon_{\max}],$$

where $d = 1, \ldots, D$. As seen, the tightness of the variable intervals depends on the value of $\epsilon_{\max}$. If $\epsilon_{\max}$ grows towards $\epsilon$, the variable intervals shall fall back to the original formulations. Nonetheless, as said, considering adversarial examples often appear closely around the original data point, we conjecture that a small region would have already contained all of them, offering a high possibility for a quick solution.

We summarize the progressive procedure in Algorithm 2. Specifically, we use Gurobi 9.5 [20] 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 $\omega$, counterexample $x'$, solving gap $\alpha$ and execution time $t_{\text{exec}}$. The status returned can be optimal (OPT), satisfied (SAT), undetermined (UNDTM) and unsatisfied (UNSAT). Additionally, to prevent the verifier from executing unboundedly, we place a time limit $t_{\text{limi}}$ on the overall algorithm. Lastly, apart from fastening the verification process, another advantage of such a progressive procedure is that one can still attain a good lower bound of the optimal objective if a timeout occurs (Algorithm 2, Line 9).

**V. EXPERIMENT RESULTS AND DISCUSSIONS**

This section presents the experimental results of the proposed method and techniques. We first introduce the Lane Departure Warning (LDW) task, then show an accuracy benchmark, and present an ablation study on the heuristics and a robustness comparison in the end.

**A. Lane Departure Warning**

We regard LDW, an industry-oriented use case, which is fundamentally a time-series joint classification and regression task [21]. For training and evaluating the NNs, we utilize

---

4This technique applies to both softmax and sparsemax.

5We write again in vector form only and omit the division over $\sqrt{D_H}$ here as it is simply a constant.
Algorithm 2 Verify with partitioning and a time limit

1: procedure VERIFYMODEL( f, x, gl(x), ε, εstep, εmin, εmax, llimit )
2: Encode f, x, gl(x) in MIQCP $\mathcal{M}$ as formulated in (10)-(13) for the sub-region of $ε_{min}$ and $ε_{max}$. 
3: Solve $\mathcal{M}$ with Gurobi under $l_{limit}$ and obtain interim results $s, n, obj, x', gap, t_{exec}$
4: if $s$ is optimal then
5:   return (OPT, $obj, x'$)
6: else if $s$ is timeout and $n > 0$ then
7:   return (SAT, $obj, x', \alpha$)
8: else if $s$ is timeout and $n = 0$ then
9:   return (UNDTM, $ε_{min}$)
10: else if $s$ is infeasible then
11:   $l_{limit} -= t_{exec}$
12:   $ε_{min} += ε_{step}$
13:   $ε_{max} = max(ε_{max} + ε_{step}, ε)$
14: if $ε_{min} > ε$ then
15:   return (UNSAT)
16: else
17:   return VERIFYMODEL( f, x, gl(x), ε, εstep, εmin, εmax, llimit )
18: end if
19: end if
20: end procedure
21: Input: f, x, gl(x), ε, εstep, llimit
22: Initialize: $ε_{min} ← 0$, $ε_{max} ← ε_{step}$
23: result = VERIFYMODEL(f, x, gl(x), ε, εstep, εmin, εmax, llimit)
24: Output: The resulting tuple result

TABLE I: Accuracy 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.

| Network | Activation | LN  | L = 1 | CLS | REG | L = 2 | CLS | REG |
|---------|------------|-----|-------|-----|-----|-------|-----|-----|
| ATN     | sparsemax  | 2   | 98.31% | 88.46% | 98.50% | 91.49% |
|         |            | 1   | 98.52% | 91.39% | 98.54% | 92.31% |
|         | softmax    | 2   | 98.01% | 85.11% | 98.33% | 92.88% |
|         |            | 1   | 97.95% | 85.06% | 98.61% | 91.52% |
|         |            | 0   | 98.21% | 88.03% | 98.64% | 91.21% |
| MLP     | ReLU       | 2   | 97.62% | 83.81% | 97.81% | 84.70% |
|         |            | 1   | 97.46% | 85.55% | 97.89% | 85.04% |
|         |            | 0   | 97.68% | 84.27% | 97.92% | 84.81% |
|         | tanh       | 2   | 97.80% | 83.20% | 97.99% | 84.93% |
|         |            | 1   | 97.32% | 82.93% | 97.69% | 84.81% |
|         |            | 0   | 97.23% | 83.02% | 97.85% | 85.14% |

MLPs, we replace MSA of the ATN with another MLP of hidden-layer dimension $D_{MLP} = 16$, resulting in similar numbers of network parameters. We implement the NNs with PyTorch [23], train them using the Adam optimizer and a fixed learning rate of 0.003 for 50 epochs, and report the best results in Table I.

As observed, even for a relatively small application (considering the variable dimensions), the ATNs generally perform better than MLPs in terms of accuracy. Additionally, networks with piece-wise linear activation functions (i.e., sparsemax and ReLU) are on a par with, if not stronger than, the ones with softmax or tanh.

C. Ablation Study

We now conduct an ablation study on the acceleration heuristics described in Section IV-C, using the sparsemax-based ATN with $L = 1$ and LN = 1. Additionally, we implement two heuristics offered by Gurobi, namely variable hints and branching priorities [20]. The former provides the solver a better foundation in searching, whereas the latter regulates the order of the integer variables selected for branching when their values are still fractional. Concretely, we run the Projected Gradient Descent (PGD) attack [24] and feed an adversarial example, if it exists, into the solver as variable hints. Then, we set integer variables stemming from the earlier layers of the NN to have higher branching priorities. These two techniques have been found helpful by the prior art [6], [25].

During our verification, 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 II. All experiments are run with an Intel i9-10980XE CPU @ 3.0GHz using 18 threads and 20 GB of RAM.

It is observed that the novel activation bounding technique is effective. Additionally, progressive verification with norm-space partitioning further reduces verification time, as expected. However, the best epsilon step might vary among test
cases as we see its magnitude does not necessarily correlate to the best-case verification speed. A reason behind this might be the excessive number of MIP instances created by smaller epsilon steps. Lastly, combining interval analysis and progressive verification delivers a speedup of approximately an order of magnitude.

D. NN Robustness Comparisons

For robustness comparisons, we first verify and compare the robustness of the sparsemax-based ATN and the ReLU-based MLP of similar sizes, which have recorded 98.31% and 97.46% in accuracy for classification in Section V-B.\(^7\) We set \(\varepsilon = 0.03\) and \(p = 1\) for the admissible \(\ell_p\)-ball\(^8\) and enable IA-\(\sigma+\)RP-0.01 from the previous section during verification. We collect results from 60 random data points (on which both the ATN and MLP predict correctly before perturbation) and plot the robustness comparison diagram in Fig. 3. We first observe that the data samples are relatively distributed across the upper and lower triangles. In addition, a dense cluster is lying in the upper left corner, indicating that the MLP demonstrates a much larger resilience than the ATN for a significant portion of the data.

We further train and verify the robustness of five ATNs and five MLPs, each on 20 data points, giving 100 samples for each NN type. For faster verification, we set \(\varepsilon = 0.01\) as the threshold for being robust and surprisingly find that all MLPs are verified robust on all data points. In contrast, the ATNs are verified robust on 69 data points and have a mean robustness value of 0.0041 (excluding the robust ones). The finding is opposed to the observations made by some related work on vision tasks [26], [27]. It is, therefore, believed that NNs generally perform differently in diverse domain tasks. As a result, it is necessary to conduct thorough studies and give rigorous guarantees on NN accuracy and robustness before deploying NN-based applications.

VI. Conclusion

This paper works towards exact robustness verification for ATNs. We focus on the sparsemax-based ATNs, encode them into a MIQCP problem, and propose accelerating heuristics for solving. When applied, our proposals fasten the verification process roughly one order of magnitude. We conduct experiments with a lane departure warning system and discover that ATNs are less robust than MLPs.

Still, there are certain limitations in our work. First, we focus only on small-scale networks (approximately 1600 neurons) and datasets (with input of 14 \(\times\) 10 dimensions). Whether our observations remain true for larger-scale networks and datasets is yet to be explored. Second, the paper only examines point-wise robustness verification for NNs. Such analyses can be combined with systematic sampling and testing methods to give formal and statistical guarantees on safety-critical applications. Lastly, our verification requires ground truths and works only in design time. How to utilize the studied techniques in a run-time setting is an open question.

\(^7\) Verifying the MLP follows similar steps in Section IV except that the encoding is relatively more straightforward and can be solved by Mixed Integer Linear Programming (MILP).

\(^8\) 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 LDW. Perturbations in this context can stem from sensor noises or hardware faults. Lastly, the binary features (i.e., lane existence) are not perturbed.

\[ \text{TABLE II: Ablation study on the considered acceleration heuristics, including interval analysis without and with sparsemax activation bounding (IA and IA-\(\sigma\)) and norm-space partitioning with different epsilon steps (RP-0.001, RP-0.005 and RP-0.01).} \]

| Techniques        | Time elapsed (s) | Nodes explored | Random samples |
|-------------------|------------------|----------------|----------------|
|                   | Mean | Best | Worst | Mean | Best | Worst | No. 1 | No. 2 | No. 3 |
| Control           | 2367.14 | 89.63 | 3620.52 | 932675 | 118329 | 16395255 | (UNDTM, 0.0) | (SAT, 1.0) | OPT   |
| IA                | 1703.65 | 330.26 | 3605.29 | 5370138 | 911639 | 13103193 | (UNDTM, 0.0) | OPT | OPT   |
| IA-\(\sigma\)     | 950.37 | 127.33 | 2936.72 | 3061462 | 247916 | 10298216 | OPT | OPT | OPT   |
| RP-0.001          | 755.66 | 7.08 | 3609.83 | 849968 | 11517 | 4058680 | (UNDTM, 0.005) | OPT | OPT   |
| RP-0.005          | 278.20 | 15.41 | 1276.86 | 276310 | 13734 | 1255818 | OPT | OPT | OPT   |
| RP-0.01           | 793.76 | 51.39 | 1556.72 | 1825979 | 70243 | 3429947 | OPT | OPT | OPT   |
| IA-\(\sigma\)+RP-0.001 | 852.67 | 5.36 | 3608.60 | 1689582 | 14557 | 6573066 | (UNDTM, 0.004) | OPT | OPT   |
| IA-\(\sigma\)+RP-0.005 | 296.90 | 7.74 | 1413.32 | 505555 | 7156 | 2413044 | OPT | OPT | OPT   |
| IA-\(\sigma\)+RP-0.01 | 222.68 | 59.56 | 644.10 | 494973 | 99642 | 1479260 | OPT | OPT | OPT   |

Fig. 3: Robustness of the ATN and MLP on random data points (better if larger). The dashed line separates regions where ATN or MLP gives a better robustness performance. 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” can be further pushed to the right if the verifier is given more time.
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] C.-H. Cheng, G. Nührenberg, and H. Ruess, “Maximum resilience of artificial neural networks,” in AVTA, 2017.

[7] V. Tjeng, K. Xiao, and R. Tedrake, “Evaluating robustness of neural networks with mixed integer programming,” in ICLR, 2019.

[8] R. Ehlers, “Formal verification of piece-wise linear feed-forward neural networks,” in AVTA, 2017.

[9] G. Katz, C. Barrett, D. Dill, K. Julian, and M. Kochenderfer, “Relu-plex: An efficient smt solver for verifying deep neural networks,” in CAV, 2017.

[10] M. Everett, G. Habibi, and J. P. How, “Robustness analysis of neural networks via efficient partitioning with applications in control systems,” IEEE Control Syst. Lett., vol. 5, pp. 2114–2119, 2021.

[11] 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.

[12] E. Wong and J. Z. Kolter, “Provably effective defenses against adversarial examples via the convex outer adversarial polytope,” in ICML, 2018.

[13] 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.

[14] 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.

[15] Z. Shi, H. Zhang, K.-W. Chang, M. Huang, and C.-J. Hsieh, “Robustness verification for transformers,” in ICLR, 2020.

[16] G. Bonaerti, D. I. Dimitrov, M. Baader, and M. Vechev, “Fast and precise certification of transformers,” in PLDI, 2021.

[17] R. Xiong, 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] A. F. T. Martins and R. F. Astudillo, “From softmax to sparsemax: A sparse model of attention and multi-label classification,” in ICML, 2016.

[19] I. E. Grossmann, “Review of nonlinear mixed-integer and disjunctive programming techniques,” Optim. Eng., vol. 3, pp. 227–252, 2002.

[20] Gurobi Optimization, LLC, “Gurobi optimizer reference manual,” 2021.

[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. Kloecker, and L. Eckstein, “The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems,” in ITSC, 2018.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in NeurIPS, 2019.

[24] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” in ICLR, 2018.

[25] Y. Zhu, F. Wang, W. Wan, and M. Zhang, “Attack-guided efficient robustness verification of relu neural networks,” in IJCNN, 2021.

[26] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit, “Understanding robustness of transformers for image classification,” in ICCV, 2021.

[27] R. Shao, Z. Shi, J. Yi, P.-Y. Chen, and C.-J. Hsieh, “On the adversarial robustness of vision transformers,” in UCCV, 2021.