Neural Network Verification using Polynomial Optimisation

Matthew Newton and Antonis Papachristodoulou

Abstract— **The desire to provide robust guarantees on neural networks has never been more important, as their prevalence in society is increasing. One popular method that has seen a large amount of success is to use bounds on the activation functions within these networks to provide such guarantees. However, due to the large number of possible ways to bound the activation functions, there is a trade-off between conservativeness and complexity. We approach the problem from a different perspective, using polynomial optimisation and real algebraic geometry (the Positivstellensatz) to assert the emptiness of a semi-algebraic set. We show that by using the Positivstellensatz, bounds on the robustness guarantees can be tightened significantly over other popular methods, at the expense of computational resource. We demonstrate the effectiveness of this approach on networks that use the ReLU, sigmoid and tanh activation functions. This method can be extended to more activation functions, and combined with recent sparsity-exploiting methods can result in a computationally acceptable method for verifying neural networks.**

I. INTRODUCTION

The development of Alexnet [1] and Resnet [2] in the past decade has facilitated a huge resurgence of interest in neural networks (NNs) and in the field of machine learning. Industrial applications of NNs are ever expanding, due to the increase in computational power available and the prevalence of big data. Examples of such areas include image recognition, weather prediction and natural language processing [3]. One important consideration is the increasing use of NNs in safety-critical applications, such as autonomous vehicle technology. This accentuates the biggest shortcoming of NNs, which is their sensitivity to adversarial inputs: small changes in the input set can lead to large changes in the output. Despite considerable effort from the research community to improve our understanding and to allow certification of NNs, to date guarantees on these NNs are not sufficient for their widespread use in safety-critical applications.

There exists work in the area of NN control dating back to the 1990s [4]. However it has been the recent success of NNs in machine and reinforcement learning applications, and the parallel progress in advanced robust control methods, that has created a desire to study problems at their intersection. By diverging from traditional model-based approaches, works from reinforcement learning have provided a bridge to develop data-driven control methods [5]. There are multiple aspects of control systems that NNs have been used for.

To overcome issues surrounding adversarial attacks, techniques exist to compute robust guarantees on the NN to determine if it is safe or not. A common method of achieving this is to place bounds on the non-linear activation functions in the NN [6]. Many results in recent years have proved this method to be successful, with a number of results providing tighter and more efficient bounds on the properties of NNs. The simplest of these methods is interval bound propagation [7], which looks at the worst-case outcome of each layer in the network. There are frameworks that provide linear bounds on the activation function, which result in a linear program that can be solved using various optimisation methods [8]. Additionally, researchers have improved the accuracy and scalability of the problem. Works from [9] focus on the scalability issue using a dual approach. By combining the constraints from multiple activation functions, [10] proposes a new parametric framework called ‘k-ReLU’. A similar method is used in [11], by considering the multivariate input space on the activation functions.

Instead of linear relaxations, semidefinite relaxations were introduced in [12] to certify the robustness properties of NNs. However, semidefinite programs (SDPs) scale worse than linear programs. To overcome this, an iterative eigenvector approach was used to improve the efficiency for large scale NNs [13]. Another semidefinite framework can be achieved by using quadratic constraints [14], which can provide tight bounds on the outputs of the network. The scalability of this framework was improved in [15], by exploiting the sparsity pattern that is intrinsic to the NN structure. This formulation can be extended to perform reachability analysis on a control feedback system [16]; a similar approach can analyse the stability of an NN controller [17].

In this paper, we approach the problem from a different perspective by considering a general class of constraints representing different activation functions, leading to a semi-algebraic set. The emptiness of this set can then be tested using Positivstellensatz (Psatz) [18] and polynomial optimisation. This method is different, as instead of considering only one type of constraint e.g. linear, quadratic etc., we can use a mixture of the constraints and easily trade-off solution accuracy with computational effort. Testing the algebraic condition on the Psatz can be done using sum of squares (SOS), or equivalently by solving an SDP. This SOS framework can be parsed in MATLAB using SOSTOOLS [19] and the resulting SDP can be solved using a relevant solver e.g. SDPT3 [20].

The first contribution in this paper shows that the Psatz formulation can drastically tighten the bounds when using the ReLU activation functions against existing methods. The second contribution is that we provide significantly tighter bounds on the sigmoid and tanh activation functions by

---

This work was supported by EPSRC grants EP/L015897/1 (to M. Newton) and EP/M002454/1 (to A. Papachristodoulou).

M. Newton and A. Papachristodoulou are with the Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, U.K. {`matthew.newton`,`antonis`}`@eng.ox.ac.uk`

using a combination of two sector constraints. Using both of these contributions allows more flexibility in the optimisation problem. This is because any set of constraints can be defined and then the accuracy may be improved if needed by increasing the order of the Psatz test.

Section II describes the NN verification problem and provides an overview of the mathematical framework. The problem formulation is delineated in Section III and then in Section IV the results are presented and discussed. This paper is concluded in Section V.

II. PRELIMINARIES

A. Neural Network Verification Problem Definition

We can describe a multi-layer feed-forward neural network (NN) as a non-linear function $\pi : \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$, where n_u is the number of inputs and n_y is the number of outputs. Consider a set of all possible inputs into the NN $\mathcal{U} \subset \mathbb{R}^{n_u}$; the NN will map these inputs to a set of outputs $\mathcal{Y} \subset \mathbb{R}^{n_y}$. The input to output mapping can be expressed as

$$\mathcal{Y} = \pi(\mathcal{U}) := \{y \in \mathbb{R}^{n_y} \mid y = \pi(u),\ u \in \mathcal{U}\}.$$

The NN verification problem asks that the output of the NN must lie within a safe region $\mathcal{S}_y$ given a set of inputs $\mathcal{U}$. Since there are no guarantees on the convexity of the $\mathcal{S}_y$ set (in fact it is likely that it is non-convex) it is computationally expensive to check if these outputs lie in the safe set. To overcome this a relaxation can be computed as a conservative approximation of the set $\mathcal{Y}$ denoted by $\hat{\mathcal{Y}}$, through checking the condition $\hat{\mathcal{Y}} \subseteq \mathcal{S}_y$.

B. Neural Network Model

Consider a feed-forward fully connected NN $\pi : \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$, with ℓ layers. This can be defined by the equations:

$$\begin{aligned} x^0 &= u, \\ v^k &= W^k x^k + b^k, \text{ for } k = 0, \dots, \ell - 1, \\ x^{k+1} &= \phi(v^k), \text{ for } k = 0, \dots, \ell - 1, \\ \pi(u) &= W^\ell x^\ell + b^\ell, \end{aligned}$$

where $W^k \in \mathbb{R}^{n_{k+1} \times n_k}$, $b^k \in \mathbb{R}^{n_{k+1}}$ are the weights matrix and biases of the $(k+1)^{th}$ layer respectively and $u = x^0 \in \mathbb{R}^{n_u}$ is the input into the network. The number of neurons in the k^{th} layer is denoted by n_k; the total number of neurons in the NN is therefore $n = \sum_{k=1}^{\ell} n_k$. The non-linear activation function ϕ is applied element-wise to the $v^k = W^k x^k + b^k$ terms such that

$$\phi(v^k) := [\phi(v_1^k), \dots, \phi(v_{n_k}^k)]^T,\ v^k \in \mathbb{R}^{n_k},$$

where ϕ is the activation function and v_j^k is the pre-activation value. There are many different activation functions such as ReLU, tanh, sigmoid, GELU and ELU. In this paper we will focus on the ReLU, sigmoid and tanh activation functions, however this work can be extended to any activation function.

C. The Positivstellensatz

The NN verification problem can be recast as a set emptiness problem. The Positivstellensatz (Psatz) [18] links the emptiness of a semi-algebraic set to an algebraic condition. We express the semi-algebraic set with notation

$$S = \{x \in \mathbb{R}^n \mid g_i(x) \geq 0,\ h_j(x) = 0 \\ \forall\, i = 1, \dots, p,\ j = 1, \dots, q\}, \quad (1)$$

where g_i and h_j are polynomial functions. Throughout we define $\mathbb{R}[x_1, \dots, x_n]$ to be the set of polynomials in $x_1, \dots, x_n$ with real coefficients. We denote $x = (x_1, \dots, x_n)$ for simplicity.

Definition 1: A polynomial $p(x)$ is a sum of squares (SOS) polynomial if and only if it can be expressed as

$$p(x) = \sum_{i=1} r_i^2(x) \equiv p(x) \text{ is SOS}.$$

We define the set of polynomials that admit this decomposition by $\Sigma[x]$.

Definition 2: The cone of a set of polynomials is

$$\text{cone}\{g_1, \dots, g_p\} = \left\{ \sum_{i=1}^{p} s_i g_i \mid s_i \in \Sigma[x], g_i \in \mathbb{R}[x] \right\}.$$

Definition 3: The ideal of a set of polynomials is

$$\text{ideal}\{h_1, \dots, h_q\} = \left\{ \sum_{j=1}^{q} t_j h_j \mid t_j \in \mathbb{R}[x] \right\}.$$

Theorem 1: (Positivstellensatz) Given a semi-algebraic set S defined in (1), the following are equivalent:

1) The set S is empty.
2) $-1 \in \text{cone}\{g_1, \dots, g_p\} + \text{ideal}\{h_1, \dots, h_q\}$.

Theorem 1 links the emptiness of a semi-algebraic set with an algebraic test. It can be reformulated into what is referred to as Schmüdgen's Positivstellensatz [21].

Theorem 2: Suppose that S defined in (1) is compact. If $f(x) > 0,\ \forall\, x \in S$ then there exist $s_i, r_{ij}, \ldots \in \Sigma[x]$ and $t_j \in \mathbb{R}[x]$ such that

$$f = 1 + \sum_{j}^{q} t_j h_j + s_0 + \sum_{i}^{p} s_i g_i + \sum_{i \neq j}^{p} r_{ij} g_i g_j + \dots.$$

D. Sum of Squares

To compute the emptiness of semi-algebraic sets using the Psatz, one can use polynomial optimisation and SOS to check the algebraic condition. This results in a set of SOS conditions, which can be checked using SOSTOOLS [19]. By choosing higher degree multipliers s_i, t_j etc. one can obtain a series of set emptiness tests of increasing complexity and non-decreasing accuracy. An example of this is how the Psatz generalises and extends the S-procedure [22] in control.

Definition 4: Given a set of symmetric matrices $\{A_k \in \mathbb{R}^{n \times n} \mid k = 0, \dots, p\}$, the S-procedure states that if $A_0 - \sum_{k=1}^{p} \lambda_k A_k \succeq 0$, where λ_k is a non-negative scalar $\forall k = 1, \dots, p$ then

$$\bigcap_{k=1}^{p} \{x \in \mathbb{R}^n \mid x^T A_k x \geq 0\} \subseteq \{x \in \mathbb{R}^n \mid x^T A_0 x \geq 0\}.$$

This can be written as the Psatz by considering if the set

$$\{x \in \mathbb{R}^n \mid x^T A_1 x \geq 0, \ldots, x^T A_p x \geq 0, -x^T A_0 x > 0\}$$

is empty. Since there are no equality constraints, the Psatz condition becomes $-1 \in \text{cone}\{g_1, \ldots, g_p\}$. By setting the inequality constraints to $g_i = x^T A_i x$ and the SOS multipliers to $s_i = \lambda_i$ one can recover the standard S-procedure test, which is known to be conservative. By increasing the order of the multipliers or by multiplying the constraints together (as in the Psatz) one can obtain better tests for the set emptiness, however these tests are more computationally expensive. Therefore, compared to the full Psatz, the S-procedure is limited. For more details on the S-procedure, the readers should refer to [22]. Previous methods that analyse the NN verification problem often use S-procedure-type arguments but these results can be improved using the general formulation of the Psatz, as we will see below.

III. PROBLEM FORMULATION

A. Formulation of Constraints

We can split the constraints into three main categories, by considering the input, hidden layer and output constraints separately. The input constraints are bounded by a hyper-rectangle defined by $\mathcal{U} = \{u \in \mathbb{R}^{n_u} \,|\, \underline{u} \leq u \leq \overline{u}\}$. Therefore, the input constraints can be written as

$$g_{in}^1 = u - \underline{u} \geq 0, \; g_{in}^2 = -u + \overline{u} \geq 0.$$

We will assume that the safe set is given by the polytope

$$\mathcal{S}_y = \bigcap_{m=1}^{M} \{y \in \mathbb{R}^{n_y} \mid c_m^T y - d_m \leq 0\},$$

where $c_m \in \mathbb{R}^{n_y}$ and $d_m \in \mathbb{R}$ are given. We consider each of the faces of the polytope in turn, and attempt to obtain a bound γ_m on each d_m to approximate the true safe set. As such, the search for the bounds (γ_m) on the safe output set can be split into M SOS programs. Therefore, the output constraints contain a decision variable that can be optimised in the SOS program such that

$$g_{out}^m = \gamma_m - c_m^T y \geq 0,$$

where γ_m is the decision variable to be optimised and m denotes the m^{th} SOS program.

The hidden layer constraints can be represented by relationships between $\phi(v^k)$ and v^k, which can be expressed through properties of the activation function. Since the activation functions are applied element-wise to the v^k terms, we consider the relationship between $\phi(v_j^k)$ and v_j^k separately. To simplify the notation, we denote the relationship between $\phi(v_j^k)$ and v_j^k as ϕ and x. These are formed through sector and slope constraints, as well as bounds computed using an efficient pre-processing step known as interval bound propagation (IBP) [7]. IBP is a zeroth order method, which uses interval arithmetic – it will find the minimum and maximum bounds on the activation function, $(\underline{\phi}, \overline{\phi})$ such that

$$\phi - \underline{\phi} \geq 0, \; -\phi + \overline{\phi} \geq 0. \tag{2}$$

The pre-activation values from IBP are denoted as $(\underline{x}, \overline{x})$.

B. ReLU Function

The ReLU function is given by

$$\text{ReLU}(x) = \phi(x) = \begin{cases} 0 & x \leq 0, \\ x & x > 0. \end{cases}$$

We can gain tight bounds on the ReLU function using two inequalities and one equality constraint [12] such that

$$\phi \geq 0, \; \phi - x \geq 0, \; \phi(\phi - x) = 0. \tag{3}$$

These constraints are shown visually in Fig. 1. The values from the IBP can be used to further tighten these constraints; please see [14] for more details.

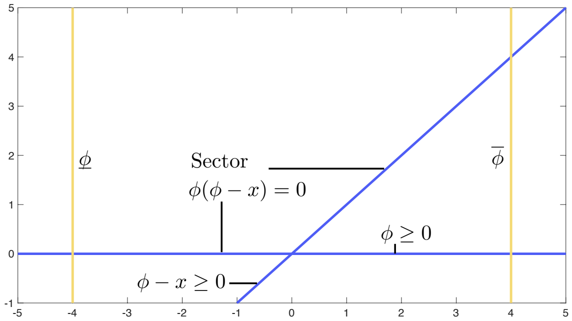


Fig. 1: Plot showing the constraints (3) that bound the ReLU function. The blue line represents the quadratic equality constraint and linear inequality constraints. The IBP values are shown in yellow.

C. Sigmoid Activation Function

The sigmoid function is given by

$$\text{sig}(x) = \phi(x) = \frac{1}{1 + e^{-x}}.$$

The simplest bound on this activation function is that the output must be in the region $\underline{\phi} \leq \phi \leq \overline{\phi}$. These constraints are of course very conservative, hence we employ sector constraints to provide better results. In this paper, we bound the sigmoid and tanh activation functions through two overlapping sectors and optimise the position and slope. Although sector conditions have been used in previous works [14], [17], they have not used multiple overlapping sectors. Previous formulations have only considered a single sector constraint, which is shown in Fig. 2a and can be written as

$$(\phi - 0.5)(0.25x + 0.5 - \phi) \geq 0. \tag{4}$$

Having two sectors will provide much tighter bounds on the activation function as it captures the point of inflection on the sigmoid and tanh functions, which is impossible to achieve with a single sector constraint. The disadvantage of our approach is that it will have twice the number of sector constraints, increasing the number of optimisation variables and therefore increase the SDP solve time.

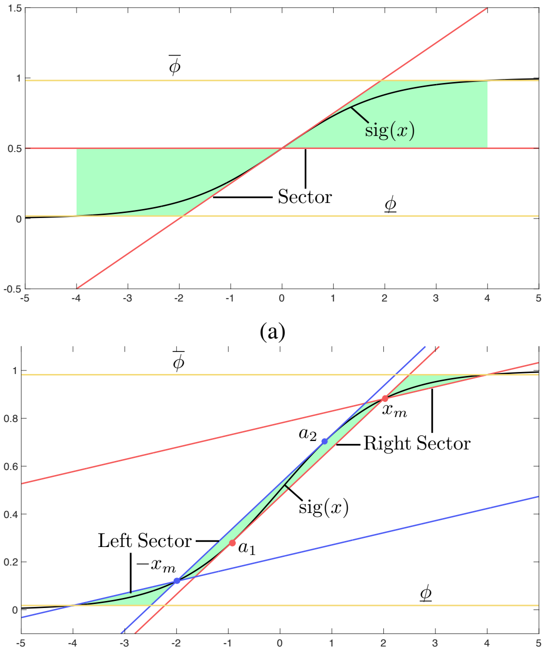
We can position the two sectors by using the IBP values to minimise the uncertainty in the constraints. The arrangement for positioning these sectors is shown visually in Fig. 2b. We start by positioning one sector at an arbitrary point on the sigmoid curve $(x_m, \phi(x_m))$. The lower line of the sector is the line passing through the points $\phi(x_m)$ and $\overline{\phi}$. The upper line of the sector passes through the point $\phi(x_m)$ and a point that is tangent to the sigmoid curve to the left of the x_m point (a_1). The equation of the lower line is

$$y(x) = \frac{\phi(x_m) - \overline{\phi}}{x_m - \overline{x}}(x - \overline{x}) + \overline{\phi}. \tag{5}$$

The gradient of the upper line can be found with the formula

$$\left.\frac{\partial \phi}{\partial x}\right|_{x=a_1} = \phi(a_1)(1 - \phi(a_1)) = \frac{\phi(x_m) - \phi(a_1)}{x_m - a_1}.$$

Given a value of x_m, this non-linear equation can be solved numerically. The same can be repeated for the sector constraint to the left of the y-axis, with the values of x_m and $\overline{\phi}$ replaced with $-x_m$ and $\underline{\phi}$ respectively. The values of x_m can be treated as hyperparameters to be chosen; we found that a value of $x_m = 1.5$ provided the best overall results in our simulations. To interpret this we can think of the uncertainty as the area between the sectors; it can be shown numerically that a value of approximately 1.5 will minimise this area on average. If the sectors overlap in the wrong region, then the slope of each sector can be adjusted accordingly.



(a)

(b)

Fig. 2: (a) Plot showing the single sector constraint (red). (b) Plot showing the two sector constraints (red and blue). Both constraints bound the sigmoid function (black). The yellow lines represent the preprocessing bounds and the area in green represents the region bound by the constraints.

D. Tanh Activation Function

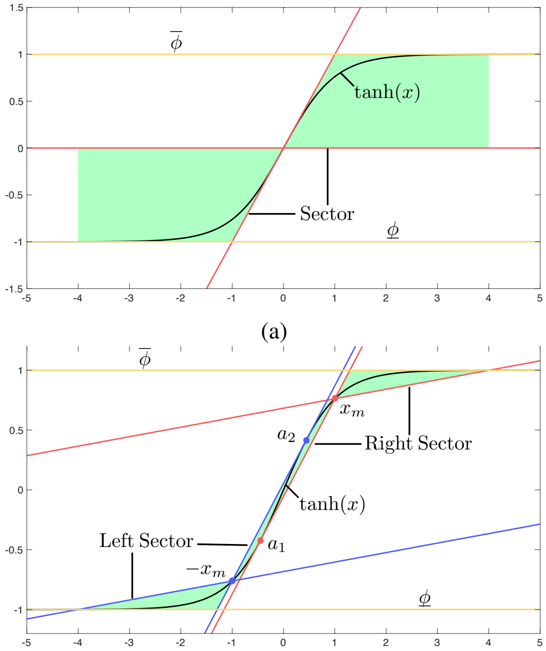
The tanh function is given by

$$\tanh(x) = \phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Similar to the sigmoid case, the function remains within the bounds from the IBP step. The process of computing the sectors is also the same. The gradient of the lower line can be calculated using the same formula as in Equation (5). However, the gradient of the upper line is replaced with a different non-linear equation

$$\left.\frac{\partial \phi}{\partial x}\right|_{x=a_1} = 1 - \tanh^2(a_1) = \frac{\phi(x_m) - \phi(a_1)}{x_m - a_1}.$$

We found that a value of $x_m = 1$ gave the best results. The single sector constraint is shown in Fig. 3a and the two sector constraint is shown in Fig. 3b.



(a)

(b)

Fig. 3: (a) Plot showing the single sector constraint (red). (b) Plot showing the two sector constraints (red and blue). Both constraints bound the tanh function (black). The yellow lines represent the preprocessing bounds and the area in green represents the region bound by the constraints.

E. Slope Constraints

Another set of constraints that can be used are obtained by considering the slope of two activation functions together. This method is employed in [14], where the slope can be bounded by a sector constraint such that

$$\alpha \leq \frac{\phi(x_2) - \phi(x_1)}{x_2 - x_1} \leq \beta.$$

Therefore, any two nodes in the NN must satisfy

$$(\phi(x_i) - \phi(x_j) - \alpha(x_i - x_j))(\beta(x_i - x_j) - (\phi(x_i) - \phi(x_j)) \geq 0,$$

$\forall i, j = 1, \ldots, n,\ i \neq j$. For the activation functions in this paper, the slope restricted sectors are $\alpha = 0$ and $\beta = 1$ for ReLU and tanh and $\alpha = 0$ and $\beta = 0.25$ for sigmoid. However, the big issue with constraints of this type is the lack of scalability. As the number of neurons in the network increases, the number of constraints increases with order $\binom{n}{2}$.

We show in Section IV through examples that constraints of this type do not increase the accuracy of the bounds as much as looking for higher order problem representations using the Psatz.

IV. NUMERICAL RESULTS

We now compare our method and existing methods for various NN sizes, with different activation functions. All experiments were run on a 4-core processor with 16GB of RAM. For our 'NNPsatz' method, we use SOSTOOLS [19] and SDPT3 [20] to parse and solve the SDP. The methods that we compare our method to are DeepSDP (without slope constraints) [14], DeepSDP$\star$ (with slope constraints), Interval Bound Propagation (IBP) [7] and the true values which are computed by exhaustive search.

We adjust the Psatz condition slightly to use it more easily in conjunction with SOSTOOLS. Instead of showing that $g_{out}^m \geq 0$ is feasible, we can show that $g_{out}^m < 0$ is infeasible using the Psatz. By setting γ_m as the decision variable in the SOS program, it can be optimised to find the limiting value to when this emptiness condition is violated. The SOS conditions are then written as:

$$-c_m^T y + \gamma_m - \sum_{j}^{q} t_j h_j - \sum_{i}^{p} s_i g_i - \sum_{i \neq j}^{p} r_{ij} g_i g_j - \ldots \text{ is SOS},$$

$$s_i \text{ is SOS}, \ \forall\, i = 1, \ldots, p, \quad r_{ij} \text{ is SOS}, \ \forall\, i, j = 1, \ldots, p,$$
$$t_j \in \mathbb{R}[x], \ \forall\, j = 1, \ldots, q,$$

where h_j and g_i are the equality and inequality constraints.

There are a large number of parameters to consider in this problem, due to the NN structure. For example, we can vary the input, hidden layer and output dimensions, as well as the size of the input space. Additionally, there are many ways that the Psatz can be formulated: the number of constraints that are multiplied together can be changed and the order of these multiplied constraints can be modified as well. Furthermore, the order of the polynomial multipliers can be controlled and the number of variables contained in the multipliers can be altered to make the SOS formulation more or less conservative. This is because adding more and higher order constraints into the SOS problem increases the number of variables in the SDP. All of the required code for this paper is available at https://github.com/MNewtonOX/nnpsatz.

A. ReLU Activation Function Example

To show the effectiveness of our method, we start with a small two layer NN with two nodes in each layer, containing ReLU activation functions, with a single input and output. We consider a large input space $[\underline{u}, \overline{u}] = [-50, 50]$, to rigorously test each method. The results for this example are shown in Table I. NNPsatz uses the ReLU representation in (3) and results in a 4^{th} order polynomial optimisation problem. These results show a significant increase in accuracy over the other methods. This is of course at the expense of computational time, however this can be improved with the trade-off of reducing the solution accuracy. The solve time can also be improved with more efficient optimisation methods, that can exploit the sparsity of the problem.

TABLE I: Comparison of verification methods for ReLU

	y_{min}	y_{max}	Solve Time (s)	SDP Size
NNPsatz	4.3000	18.0384	1.1147	340×100
DeepSDP	-10.6868	69.6374	0.0767	49×18
DeepSDP$\star$	0.5371	18.0384	0.0975	55×24
IBP	-14.7009	105.3742	0.0011	-
True values	4.3000	18.0384	-	-

B. Sigmoid and Tanh Activation Function Examples

In this example we show the effectiveness of this method in conjunction with the sigmoid and tanh activation functions. We consider a single input, single output NN with four hidden layers and four nodes in each layer. We consider the input space $[\underline{u}, \overline{u}] = [0.25, 1.75]$. Since at the time of writing DeepSDP does not have an implementation for sigmoid and tanh functions, we compare the accurate NNPsatz method against similar formulations, to show how solution accuracy and computation can be traded-off. The results for the sigmoid and tanh examples are shown in Table II and Table III respectively. NNPsatz (1) uses the two sector constraints as in Section III-C and III-D and the IBP constraints (2). NNPsatz (2) is similar, however the two sector constraints are replaced by single sectors. NNPsatz (3) is the same as NNPsatz (2) with added slope restricted constraints. NNPsatz (2) and NNPsatz (3) reflect what is possible with traditional SDP formulations.

TABLE II: Comparison of verification methods for sigmoid

	y_{min}	y_{max}	Solve Time (s)	SDP Size
NNPsatz (1)	0.7417	0.7617	0.7650	360×171
NNPsatz (2)	0.6443	0.8653	0.4397	358×171
NNPsatz (3)	0.6742	0.8565	0.7658	478×171
IBP	0.6440	0.8690	0.004	-
True values	0.7465	0.7550	-	-

TABLE III: Comparison of verification methods for tanh

	y_{min}	y_{max}	Solve Time (s)	SDP Size
NNPsatz (1)	-3.8258	0.6841	0.4739	369×171
NNPsatz (2)	-4.2631	4.0443	0.4096	358×171
NNPsatz (3)	-4.2601	2.8704	0.7470	478×171
IBP	-4.2645	4.2780	0.004	-
True values	-2.8080	-2.3531	-	-

C. Two Dimensional Input and Output Example

We now consider the effectiveness for an NN with two inputs and two outputs. The hidden layer dimensions are $[3, 4, 3]$, containing ReLU activation functions. We compare NNPatz, DeepSDP and IBP for an input space $[\underline{u}, \overline{u}] = [-50, 50]$. In Fig. 4, we see that NNPsatz gives significantly tighter bounds over the other methods.

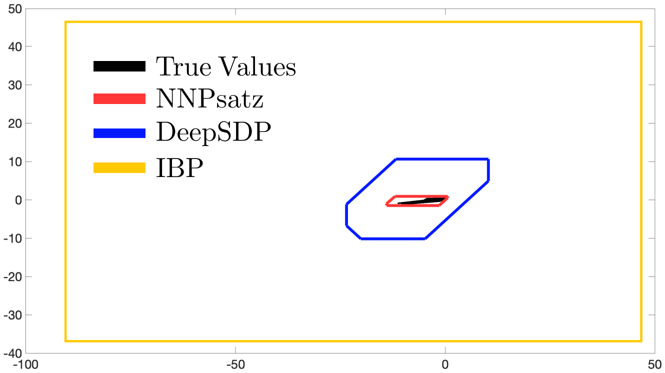


Fig. 4: Plots of the true output set (black) and the bounds that are created using NNPsatz (red), DeepSDP (blue) and IBP (yellow).

V. CONCLUSION

In this paper we have investigated the problem of providing bounds on the outputs of NNs, to ensure that they operate safely and reliably. We have approached the problem from a general framework that combines real algebraic geometry with polynomial optimisation, by setting up the problem as the emptiness of an appropriately constructed semi-algebraic set. As the constraints within this framework can be defined generally and hence can be made more or less conservative, it is possible to easily trade off solution accuracy with the amount of computational time required.

We use existing constraints on the ReLU activation function and propose a bound consisting of two sector constraints for the sigmoid and tanh activation functions. These constraints are then parsed into SOSTOOLS, which converts the problem into an SDP, which is solved using the solver 'SDPT3'. We show for many examples that this formulation can provide the tightest bounds on the NN compared to other SDP formulations. This approach is also very versatile as the constraints can be modified and a different accuracy can be set depending on the size of the network.

There is a lot of scope for future work surrounding this area. One of the biggest issues with the SDP framework is its scalability, however there are many ways of improving this. NN pruning [23] can be used to reduce the size of the NN to decrease the number of constraints in the optimisation problem. Related works in [24] synthesise a reduced order NN with robustness guarantees. Other approaches to improve the scalability include previous works from chordal sparsity [15], first-order solvers such as CDCS [25] and boosting methods.

Lastly, the examples in this paper have been focused on randomly generated NNs. It would be interesting to apply this method to real-life examples or the case of NN controllers to compare the performance. We have also only focused on ReLU, sigmoid and tanh activation functions; as mentioned earlier, the ideas from this paper can easily be applied to other activation functions. With a combination of these improvements, NNs can be verified more effectively in the future.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097-1105, 2012.

[2] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, 2016.

[3] C. Zhang and Y. Ma, *Ensemble Machine Learning: Methods and Applications*. Springer, 2012.

[4] W. T. Miller, R. S. Sutton, and P. J. Werbos, *Neural Networks for Control*. MIT Press, 1990.

[5] B. Recht, "A Tour of Reinforcement Learning: The View from Continuous Control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 253–279, 2019.

[6] H. Salman, G. Yang, H. Zhang, C. J. Hsieh, and P. Zhang, "A convex Relaxation Barrier to Tight Robustness Verification of Neural Networks," arXiv:1902.08722, 2020.

[7] S. Gowal, C. Qin, J. Uesato, and T. Mann, "On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models", arXiv:1810.12715, 2018.

[8] R. Bunel, I. Turkaslan, P. H. S. Torr, P. Kohli, and M. P. Kumar, "A Unified View of Piecewise Linear Neural Network Verification," arXiv:1711.00455, 2017.

[9] K. Dvijotham, R. Stanforth, S. Gowal, T. Mann, and P. Kohli, "A Dual Approach to Scalable Verification of Deep Networks," arXiv:1803.06567, 2018.

[10] G. Singh, R. Ganvir, M. Püschel, and M. Vechev, "Beyond the Single Neuron Convex Barrier for Neural Network Certification," *Advances in Neural Information Processing Systems*, vol. 32, pp. 14–16, 2019.

[11] C. Tjandraatmadja, R. Anderson, J. Huchette, W. Ma, K. Patel, and J. P. Vielma, "The Convex Relaxation Barrier, Revisited: Tightened Single-Neuron Relaxations for Neural Network Verification," arXiv:2006.14076, 2020.

[12] A. Raghunathan, J. Steinhardt, and P. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," *32nd Conference on Neural Information Processing Systems*, vol. 2018-Decem, pp. 10877–10887, 2018.

[13] S. Dathathri et al., "Enabling Certification of Verification-agnostic Networks via Memory-efficient Semidefinite Programming," *Advances in Neural Information Processing Systems 33*, pp. 5318–5331, 2020.

[14] M. Fazlyab, M. Morari and G. J. Pappas, "Safety Verification and Robustness Analysis of Neural Networks via Quadratic Constraints and Semidefinite Programming," *IEEE Transactions on Automatic Control*, 2020.

[15] M. Newton and A. Papachristodoulou, "Exploiting Sparsity of Neural Network Verification," *3rd Annual Learning for Dynamics and Control Conference*, 2021.

[16] H. Hu, M. Fazlyab, M. Morari and G. J. Pappas, "Reach-SDP: Reachability Analysis of Closed-Loop Systems with Neural Network Controllers via Semidefinite Programming," *59th IEEE Conference on Decision and Control (CDC)*, pp. 5929-5934, 2020.

[17] H. Yin, P. Seiler, and M. Arcak, "Stability Analysis using Quadratic Constraints for Systems with Neural Network Controllers," arXiv:2006.07579, 2020.

[18] G. Stengle, "A Nullstellensatz and a Positivstellensatz in Semialgebraic Geometry," *Mathematische Annalen 207*, no. 2, pp. 87–97, 1974.

[19] A. Papachristodoulou, J. Anderson, G. Valmorbida, S.Prajna, P. Seiler, and P. A. Parrilo, *SOSTOOLS: Sum of Squares Optimization Toolbox for MATLAB*, 2013.

[20] K.C. Toh, M.J. Todd, and R.H. Tutuncu, "SDPT3 - a MATLAB Software Package for Semidefinite Programming," *Optimization Methods and Software*, vol. 11, pp. 545–581, 1999.

[21] G. Stengle, "Complexity Estimates for the Schmüdgen's Positivstellensatz," *Journal of Complexity*, vol. 12, no. 2, pp. 167–174, 1996.

[22] V. Yakubovich, "S-procedure in nonlinear control theory," Vestnick Leningrad Univ. Math., vol. 4, pp. 73–93, 1997.

[23] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the State of Neural Network Pruning?," arXiv:2003.03033, 2020.

[24] R. Drummond, M. C. Turner, and S. R. Duncan, "Reduced-Order Neural Network Synthesis with Robustness Guarantees," arXiv:2102.09284, 2021.

[25] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn, "Chordal Decomposition in Operator-splitting Methods for Sparse Semidefinite Programs," *Mathematical Programming*, vol. 180, no. 1–2, pp. 489–532, 2020.