Convex neural network synthesis for robustness in the 1-norm

Ross Drummond ROSS.DRUMMOND@SHEFFIELD.AC.UK
Dept. of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, S1 3JD, UK.

Chris Guiver C.GUIVER@NAPIER.AC.UK
School of Computing, Engineering & the Built Environment, Edinburgh Napier University, Edinburgh, EH10 5DT, UK.

Matthew C. Turner M.C.TURNER@SOTON.AC.UK
School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK.

Editors: A. Abate, K. Margellos, A. Papachristodoulou

Abstract

With neural networks being used to control safety-critical systems, they increasingly have to be both accurate (in the sense of matching inputs to outputs) and robust. However, these two properties are often at odds with each other and a trade-off has to be navigated. To address this issue, this paper proposes a method to generate an approximation of a neural network which is certifiably more robust. Crucially, the method is fully convex and posed as a semi-definite programme. An application to robustifying model predictive control is used to demonstrate the results. The aim of this work is to introduce a method to navigate the neural network robustness/accuracy trade-off.

Keywords: Neural network robustness, convex synthesis, accuracy *vs.* robustness trade-off.

1. Introduction

Neural networks have emerged as powerful and flexible nonlinear function approximators for mapping input/output data. For control applications, the main strengths of neural networks are, arguably, their ability to learn complex feedback policies for control problems which are challenging to formulate mathematically. Examples include problems where only data is available (and no model exists) or when the costs/models/constraints may be unknown, as encountered, for instance, in image-based control of autonomous robotics. However, neural networks also have weaknesses. They suffer from a lack of robustness and explainability; two issues which have, so far, prevented their widespread adoption into safety-critical systems. As a consequence, whilst neural networks have thrived as control systems for technologies where failure is not catastrophic (such as robotic demonstrations in the lab), there remains scepticism about their value for systems where crashes can be deadly (such as with aircraft autopilot software). For these safety-critical applications, traditional control theory still dominates.

The growing appreciation of the need to embed robustness more deeply into neural network design mirrors the concerns raised by the control community during the 1980s in light of high profile incidents, such as the Gripen JAS39 prototype incident detailed in Stein (2003). In the wake of these incidents, robust control theory emerged as a new design paradigm that placed as much emphasis on a control system's robustness as its performance (Green and Limebeer (2012)), with the control synthesis results building upon earlier robustness analysis results such as the Zames-Falb

multipliers of Zames and Falb (1968). Arguably, a similar transition has yet to take place with neural networks, even after several well-publicised incidents, such as deadly autonomous vehicle crashes. The question then arises as to whether the strengths of neural networks (namely, as powerful, yet simple to train, function approximattors) can be balanced against their weaknesses (e.g. their lack of tight robustness guarantees) in a rigorous and quantifiable way.

Contribution: Addressing this balance between neural network robustness and accuracy is the focus of this paper. Specifically, a method to address the following neural network design problem is developed:

> "*Given a neural network, generate another one which is quantifiably more robust yet has a similar input/output mapping as the original neural network.*"

The solution to this problem presented here in Theorem 7 is referred to as a *neural network synthesis method*, as it is motivated by the linear matrix inequalities (LMIs) widely used in robust control synthesis Dullerud and Paganini (2013). Crucially, the proposed method is fully convexified in the sense that the weights and biases of the robustified network are obtained from the solutions of a semi-definite programme (SDP) that trade-off robustness against accuracy. This convexification is achieved by: *i*) using the classical slope restrictions on the activation functions and, more crucially, *ii*) using the 1-norm for the robustness bounds instead of the 2-norm commonly used with Lipschitz bound-type results. We demonstrate the validity of our results in an application towards robustifying model predictive control (MPC) policy.

Literature: The results of Pauli et al. (2021b) and Wang and Manchester (2023) are perhaps the most relevant to this paper. Focussing on Pauli et al. (2021b), a method to embed robustness (imposed by LMI constraints) into the neural network training was developed. This was achieved by using the ADMM algorithm to switch between training on the data and imposing Lipschitz bounds. A similar approach was developed in Wang and Manchester (2023), with points of differences being the focus on equilibrium networks and the direct parameterisation of the weights to satisfy Lipschitz bounds. One issue with the approach of Pauli et al. (2021b) is that, because robustness is measured using the 2-norm (as in Lipschitz bounds), then a bilinearity appears in the training process, as both the neural network weights/biases and the multipliers of the robustness LMIs are decision variables for the solver. This bilinearity is a source of non-convexity for the optimisation problem, and has appeared in earlier synthesis results such as Drummond et al. (2022b) on generating reduced-order approximations of large neural networks. By contrast, here we propose formulating the robustness bounds in terms of the 1-norm instead of the 2-norm. Moreover, it is proposed to approximate a given neural network which is assumed to give a good model of the data. Theorem 7 shows that with these two tweaks to the problem formulation, the trade-off between network robustness and accuracy can be fully convexified and combined within a single SDP- instead of the iterative projection type approach of ADMM.

There are deep connections between control theory and neural network robustness analysis, most notably through the application of results from absolute stability theory. Early results in this area include Chu and Glover (1999a) and Chu and Glover (1999b) where the application to neural networks led to the development of a new set of static multipliers for nonlinearities with repeated terms. Similar results were obtained by Barabanov and Prokhorov (2002) for recurrent neural networks. More recent efforts in this direction include the works of Wang et al. (2022), Fazlyab et al. (2019) and Pauli et al. (2021a) where the connection to LMIs and SDP solvers has been made more explicit. One of the major issues when using these SDP formulations is the lack of algorithm

scalability to the large networks often seen in practice. Efforts to address this scalability issue are now under development, for example with Wang et al. (2024) and Newton and Papachristodoulou (2023) where sparsity of the neural network was exploited. With the connections between neural networks maturing, there is now a push towards moving away from solving analysis type problems of neural networks and going towards controller synthesis, as exemplified by papers such as Furieri et al. (2022), Junnarkar et al. (2022) and Newton and Papachristodoulou (2022).

Notation

Real matrices M of size $m \times n$ are denoted $M \in \mathbb{R}^{n\times m}$. The set of positive-definite matrices $M \succ 0$ of size n are $M \in \mathbb{S}^n_{\succ 0}$. The set of diagonal matrices of dimension n with positive elements are $\mathbb{D}^n_+$. The identity matrix of dimension n is I_n. The set of symmetric matrices A of dimension n are $A \in \mathbb{S}^n$. The Hermitian operation is defined such that $He(M) = M + M^\top$. When appropriate, element (i, j) of a matrix M is referred to as $M^{i,j}$. The $\star$-notation for symmetric matrices is used, as in $M = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} a & b \\ \star & c \end{bmatrix}$. Real vectors p of dimension n are denoted $p \in \mathbb{R}^n$, non-negative vectors are $p \in \mathbb{R}^n_+$ and non-negative scalars are $p \in \mathbb{R}_+$. The vector of ones of dimension n is $\mathbf{1}_n$ and the vector of zeros of dimension n is 0_n. The $m \times n$ matrix of zeros is $0_{m\times n}$ and that of ones is $1_{m\times n}$. Nonlinear functions $\phi : \mathbb{R}^n \to \mathbb{R}^n$ act component-wise on their arguments, *i.e.* $\phi(s) = [\phi_1(s_1), \ldots, \phi_n(s_n)]^\top$ for $s = [s_1, \ldots, s_n]^\top \in \mathbb{R}^n$.

2. Problem setup

2.1. The original neural network

Consider a neural network $f(u) : \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$ in the implicit form of El Ghaoui et al. (2021)

$$x = \phi(W_x x + W_u u + b) \tag{1a}$$
$$f(u) = W_{f,x} x + W_{f,u} u + b_f, \tag{1b}$$

with weights $W_x \in \mathbb{R}^{n\times n}$, $W_u \in \mathbb{R}^{n\times n_u}$, $W_{f,x} \in \mathbb{R}^{n_g\times n}$, $W_{f,u} \in \mathbb{R}^{n_g\times n_u}$ and biases $b \in \mathbb{R}^n$, $b_f \in \mathbb{R}^{n_g}$ and activation functions $\phi(\cdot)$.

The primary reasons for using the implicit form of El Ghaoui et al. (2021) to represent the networks of (1) in this paper are simply: *i*) it allows a wide class of network architectures (such as recurrent and feed-forward neural networks, but also many others as detailed in El Ghaoui et al. (2021)) to be represented within a unified framework, *ii*) it makes the notation more compact. However, we expect that, after a minor tweaking of the notation, the presented results could be applied to a wider class of architectures besides those with the structure of (1). Finally, it is remarked that the implicit neural networks of (1) are structured similarly to others from the literature, most notably deep equilibrium neural networks Revay et al. (2023); Bai et al. (2019); Pokle et al. (2022).

In this work, the neural network of (1) is regarded as the true mapping of the data, and it is around this mapping that the accuracy of the robustified network is defined. Several standard assumptions are imposed on the neural network of (1).

Assumption 1 The neural network $f(u)$ of (1) is assumed to be well-posed, in the sense that for every $u \in \mathbb{R}^{n_u}$ there is a unique solution x to the implicit equation (1a).

Assumption 2 The activation function $\phi(\cdot)$ is diagonal and $[0, 1]$-slope-restricted.

Recall that a function $\phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is called diagonal if $(\phi(s))_i = \phi_i(s_i)$ for all $s \in \mathbb{R}^n$ and $i = 1, 2 \ldots, n$. A diagonal function satisfies a $[\delta_1, \delta_2]$-slope restriction, for given $\delta_1 < \delta_2$, if

$$\delta_1 \leq \frac{\phi_i(s_1) - \phi_i(s_2)}{s_1 - s_2} \leq \delta_2 \quad \forall\, s_1,\ s_2 \in \mathbb{R},\ s_1 \neq s_2\,. \tag{2}$$

The slope restriction assumption for the nonlinearities of neural networks is widely used, e.g. in Chu and Glover (1999a); Barabanov and Prokhorov (2002); Fazlyab et al. (2019); Drummond et al. (2022a). The present $[0, 1]$-slope restriction may be relaxed to a $[0, \delta]$-slope restriction for $\delta > 0$ by invoking a loop-shifting argument.

2.2. The robustified neural network and robustness in the 1-norm

The goal of this paper is to develop a method to compute a quantifiably robust neural network (in the sense of Definition 2 for some γ, $\gamma_{u,1}$, $\gamma_{u,2}$) which can approximate the input/output map of the original one given by (1). To this end, a second neural network is introduced

$$z = \phi(\Psi_z z + \Psi_u u + \beta), \tag{3a}$$

$$g(u) = \Psi_{g,z} z + \Psi_{g,u} u + \beta_g, \tag{3b}$$

with weights $\Psi_z \in \mathbb{R}^{n \times n}$, $\Psi_u \in \mathbb{R}^{n \times n_u}$, $\Psi_{g,z} \in \mathbb{R}^{n_g \times n}$, $\Psi_{g,u} \in \mathbb{R}^{n_g \times n_u}$ and biases $\beta \in \mathbb{R}^n$, $\beta_g \in \mathbb{R}^{n_g}$. The goal is to compute values for these weights and biases whilst navigating the accuracy/robustness trade-off with respect to the original neural network (1). To facilitate the robustness analysis, the following notation will be used to characterise the outputs generated by two inputs u_1 and u_2 acting on this network. Specifically, we let $(z, u) = (z_i, u_i)$ for $i = 1, 2$ denote two input/state pairs of (3), and set $g_i := g(u_i)$ as the corresponding outputs. The following set will be used to define the space of these two inputs.

Definition 1 *For some $\varepsilon_{u,1} \geq 0$, $\varepsilon_{u,2} \geq 0$, two inputs u_1 and u_2 are constrained by the set* $\mathcal{U}(\varepsilon_1, \varepsilon_2) := \{(u_1, u_2) \,:\, \|u_1 - u_2\|_2^2 \leq \varepsilon_{u,2},\ \|u_1 - u_2\|_1 \leq \varepsilon_{u,1}\}$.

In this work, the following notion of neural network robustness will be used. Compared to earlier work on neural network robustness certificates using SDPs where the 2-norm of the outputs was bounded, such as Fazlyab et al. (2019), the 1-norm is used presently.

Definition 2 *The neural network (3) is said to be robust if there exists $\gamma \geq 0$, $\gamma_{u,1} \geq 0$, $\gamma_{u,2} \geq 0$ such that*

$$\|g(u_1) - g(u_2)\|_1 \leq \gamma + \gamma_{u,1}\|u_1 - u_2\|_1 + \gamma_{u,2}\|u_1 - u_2\|_2^2 \quad \forall\, (u_1,\, u_2) \in \mathcal{U}. \tag{4}$$

The following measure of the similarity between two neural networks (Li et al. (2021)), as in their accuracy with respect to each other's input/output mappings, will be used.

Definition 3 *The two neural networks (1) and (3) are said to be similar if they have the same biases, as in $\beta = b, \beta_f = b_f$, and their weights are "close" in the sense that there exists non-negative scalars $\varepsilon_{W_x}, \varepsilon_{W_u}, \varepsilon_{W_{f,x}}$, and $\varepsilon_{W_{f,u}}$ such that, for each matrix element i, j,*

$$\begin{cases} |W_x^{i,j} - \Psi_z^{i,j}| \leq \varepsilon_{W_x}, & |W_u^{i,j} - \Psi_u^{i,j}| \leq \varepsilon_{W_u}, \\ |W_{f,x}^{i,j} - \Psi_{g,z}^{i,j}| \leq \varepsilon_{W_{f,x}}, & |W_{f,u}^{i,j} - \Psi_{g,u}^{i,j}| \leq \varepsilon_{W_{f,u}}. \end{cases} \tag{5}$$

If the above holds, then it said that $g(u) \approx f(u)$ for all $u \in \mathbb{R}^{n_u}$.

Whilst there are limitations with the above definition for network similarity (a more robust version such as that considered in Li et al. (2021) may provide improved theoretical nuance), it was observed that this form empirically performs well and, crucially, allows the problem to be convexified.

2.3. Problem statement

Solutions to the following problem are the considered.

Problem 1 Determine weights and biases of the neural network (3) such that it is robust in the sense of Definition 2 and is similar to the original neural network in the sense of Definition 3.

The idea of this problem is to explore the trade-off between neural network accuracy and robustness. The main benefit of the present results are that they lead to a convex synthesis problem.

3. Preliminary results

We present a solution to Problem 1 in Theorem 7 below, the proof of which is supported by the preliminary material in this section. This result will rely upon the expression of the 1-norm of a vector as a sum of ReLU functions. Recall that the ReLU function $r(\cdot) : \mathbb{R}^{n_\sigma} \to \mathbb{R}^{n_\sigma}$ is the diagonal function with equal components given by

$$r(\sigma) = \text{ReLU}(\sigma) = \max\left\{0, \sigma\right\} \quad \forall\, \sigma \in \mathbb{R}\,. \tag{6}$$

This nonlinearity is $[0, 1]$-slope-restricted and also satisfies a range of other quadratic constraints, as detailed in Richardson et al. (2023) for example. The key idea behind the robust neural network synthesis of Theorem 7 is the following representation of the absolute value as ReLU functions:

$$|\sigma| = \text{ReLU}(\sigma) + \text{ReLU}(-\sigma) \quad \forall\, \sigma \in \mathbb{R}. \tag{7}$$

This representation allows the 1-norm of the robustified neural network's output to be expressed as a *linear* sum of [0,1]-slope-restricted nonlinear functions acting on the output. With this linearity, the problem can be convexified. By contrast, using the 2-norm of the Lipschitz bounds introduces a bi-linearity into the matrix inequalities, as also encountered in Drummond et al. (2022b), and so only gives locally optimal results.

With $r(s) = \text{ReLU}(s)$ as in (6), define the incremental variables

$$\left\{\begin{array}{l} \begin{bmatrix}\tilde{g}\\ \tilde{u}\\ \tilde{z}\end{bmatrix} = \begin{bmatrix}g_2 - g_1\\ u_2 - u_1\\ z_2 - z_1\end{bmatrix} = \begin{bmatrix}\Psi_{g,z}\tilde{z} + \Psi_{g,u}\tilde{u}\\ u_2 - u_1\\ z_2 - z_1\end{bmatrix},\ \tilde{g}_\pm = \begin{bmatrix}r(\tilde{g})\\ r(-\tilde{g})\end{bmatrix}, \\ \tilde{u}_\pm = \begin{bmatrix}r(\tilde{u})\\ r(-\tilde{u})\end{bmatrix}, \quad \text{and} \quad p(u_1, u_2) = \begin{bmatrix}\tilde{g}_\pm^\top & \tilde{u}_\pm^\top & \tilde{z}^\top & \tilde{u}^\top & 1\end{bmatrix}^\top. \end{array}\right. \tag{8}$$

Recall that the inputs u_1, u_2 and outputs $g_1 = g(u_1)$, $g_2 = g(u_2)$ are given by the neural network (3). With these variables, the following lemmas define incremental quadratic constraints for (3).

Lemma 4 *For all $\mathbb{T}_z \in \mathbb{D}_+^n$, it follows that*

$$s_z(u_1, u_2) := p(u_1, u_2)^\top \Omega_z(\mathbb{T}_z) p(u_1, u_2) \geq 0 \quad \forall\, (u_1,\, u_2) \in \mathcal{U}\,,$$

with the matrix $\Omega_z(\mathbb{T}_z)$ defined in equation (16) in the Appendix.

Proof The quadratic constraint of $s_z(u_1, u_2)$ can be expanded out as $\tilde{z}^\top \mathbb{T}_z(\Psi_z \tilde{z} + \Psi_u \tilde{u} - \tilde{z}) \geq 0$ whose non-negativity follows from the assumed $[0, 1]$-slope-restriction of the activation function. ■

Lemma 5 *For all $\mathbb{T}_{u,1} \in \mathbb{R}_+$ and $\mathbb{T}_{u,2} \in \mathbb{R}_+$, it follows that*

$$s_u(u_1, u_2) := p(u_1, u_2)^\top \Omega_u(\mathbb{T}_{u,1}, \mathbb{T}_{u,2}) p(u_1, u_2) \geq 0 \quad \forall\, (u_1,\, u_2) \in \mathcal{U}\,,$$

with the matrices $\Omega_u(\mathbb{T}_{u,1}, \mathbb{T}_{u,2})$ defined in equation (17) in the Appendix.

Proof Using the decomposition of the 1-norm in terms of the ReLU of (7), then the quadratic constraint of $s_u(u_1, u_2)$ can be expanded out as

$$\begin{aligned} p(u_1, u_2)^\top \Omega_u(\mathbb{T}_{u,1}, \mathbb{T}_{u,2}) p(u_1, u_2) &= \mathbb{T}_{u,1}\varepsilon_{u,1} + \mathbb{T}_{u,2}\varepsilon_{u,2} - \frac{1}{2}\mathbb{T}_{u,2}\|\tilde{u}\|_2^2 \\ &\quad - \frac{1}{2}\mathbb{T}_{u,2}(r(\tilde{u})^2 + r(-\tilde{u})^2) - \mathbb{T}_{u,1}(r(\tilde{u}) + r(-\tilde{u})), \\ &= \mathbb{T}_{u,1}\varepsilon_{u,1} + \mathbb{T}_{u,2}\varepsilon_{u,2} - \mathbb{T}_{u,2}\|\tilde{u}\|_2^2 - \mathbb{T}_{u,1}\|\tilde{u}\|_1 \geq 0, \end{aligned}$$

since $(u_1,\, u_2) \in \mathcal{U}$ following Definition 1. ■

Lemma 6 *For all $\mathbb{T}_g \in \mathbb{D}_+^{n_g}$, it follows that*

$$s_g(u_1, u_2) := p(u_1, u_2)^\top \Omega_g(\mathbb{T}_g) p(u_1, u_2) \geq 0 \quad \forall\, (u_1,\, u_2) \in \mathcal{U}\,,$$

with the matrix $\Omega_g(\mathbb{T}_g)$ defined in equation (18) in the Appendix.

Proof Similarly to Lemma 4, the quadratic constraint of $s_g(u_1, u_2)$ can be expanded out as $r(\tilde{g})^\top\, \mathbb{T}_g(\tilde{g} - r(\tilde{g})) + r(-\tilde{g})^\top\, \mathbb{T}_g(-\tilde{g} - r(-\tilde{g})) \geq 0$ which is again non-negative owing to the [0,1] slope restriction of the ReLU$(\cdot)$ function. ■

4. Main result

We state the main result of the present work, a theorem containing a SDP to solve Problem 1.

Theorem 7 *Set the weights $w_0, w_1, w_2 \in \mathbb{R}_+$ and the tolerances $\varepsilon_{W_x}, \varepsilon_{W_u}, \varepsilon_{W_{f,x}}, \varepsilon_{W_{f,u}} \in \mathbb{R}_+$. If there exists some $\mathbb{T}_z \in \mathbb{D}_+^n$, $\mathbb{T}_g \in \mathbb{D}_+^{n_g}$, $\mathbb{T}_{u_1} \in \mathbb{R}_+$, $\mathbb{T}_{u_2} \in \mathbb{R}_+$, $\mathbb{Y}_z \in \mathbb{R}^{n\times n}$, $\mathbb{Y}_u \in \mathbb{R}^{n\times n_u}$, $\mathbb{Y}_{g,z} \in \mathbb{R}^{n_g\times n}$, $\mathbb{Y}_{g,u} \in \mathbb{R}^{n_g\times n_u}$, $\gamma \geq 0$, $\gamma_{u,1} \geq 0$ and $\gamma_{u,2} \geq 0$ that solves*

$$\min\, w_0\gamma + w_1\gamma_{u_1} + w_2\gamma_{u,2}, \tag{9a}$$

$$\text{subject to: } \check{\Omega}_z(\cdot) + \check{\Omega}_g(\cdot) + \Omega_u(\cdot) + \Omega_\gamma(\cdot) \prec 0, \tag{9b}$$

*with the matrices $\check{\Omega}_z(\mathbb{T}_z, \mathbb{Y}_z, \mathbb{Y}_u)$, $\check{\Omega}_g(\mathbb{T}_g, \mathbb{Y}_{g,z}, \mathbb{Y}_{g,u})$, and $\Omega_\gamma(\gamma, \gamma_{u,1}, \gamma_{u,2})$ defined in (19)–(21), and the following **element-wise** matrix inequalities hold*

$$-\varepsilon_{W_x}\mathbb{T}_z 1_{n\times n} \leq \mathbb{Y}_z - \mathbb{T}_z W \leq \varepsilon_{W_x}\mathbb{T}_z 1_{n\times n}, \tag{10a}$$

$$-\varepsilon_{W_u}\mathbb{T}_z 1_{n\times n_u} \leq \mathbb{Y}_u - \mathbb{T}_z W_u \leq \varepsilon_{W_u}\mathbb{T}_z 1_{n\times n_u}, \tag{10b}$$

$$-\varepsilon_{W_{f,x}}\mathbb{T}_g 1_{n_g\times n} \leq \mathbb{Y}_{g,z} - \mathbb{T}_g W_{f,x} \leq \varepsilon_{W_{f,x}}\mathbb{T}_g 1_{n_g\times n}, \tag{10c}$$

$$-\varepsilon_{W_{f,u}}\mathbb{T}_g 1_{n_g\times n_u} \leq \mathbb{Y}_{g,u} - \mathbb{T}_g W_{f,u} \leq \varepsilon_{W_{f,u}}\mathbb{T}_g 1_{n_g\times n_u}, \tag{10d}$$

then, the neural network of $g(u)$ with weights and biases defined by

$$\Psi_z = {\mathbb{T}_z}^{-1}\mathbb{Y}_z, \Psi_u = {\mathbb{T}_z}^{-1}\mathbb{Y}_u, \Psi_{g,z} = {\mathbb{T}_g}^{-1}\mathbb{Y}_{g,z}, \Psi_{g,u} = {\mathbb{T}_g}^{-1}\mathbb{Y}_{g,u}, \beta = b, \beta_f = b_f, \tag{11}$$

is robust in the sense of Definition 2 and similar to $f(u)$ from (1) in the sense of Definition 3.

Proof Multiplying the matrix inequality of (9b) on the right by $p(u_1, u_2)$ and on the left by its transpose implies, with the factorisation of the weights in (11), that

$$\begin{aligned}\|g(u_1) - g(u_2)\|_1 - \gamma - \gamma_{u,1}\|u_1 - u_2\|_1 - \gamma_{u,2}\|u_1 - u_2\|_2^2 + s_z(u_1, u_2) \\ + s_g(u_1, u_2) + s_u(u_1, u_2) \leq 0\,.\end{aligned} \tag{12}$$

Using Lemmas 4, 5 & 6, then the following quadratic inequalities are non-negative $s_z(u_1, u_2)$, $s_g(u_1, u_2)$, $s_u(u_1, u_2) \geq 0$ for all $(u_1, u_2) \in \mathcal{U}$. With these conditions, then (12) implies that the robustness bound of (4) holds. To show the norm conditions of (5), consider the first set of element-wise inequalities (10a)

$$-\varepsilon_{W_x}\mathbb{T}_z^{i,i} \leq \mathbb{Y}_z^{i,j} - \mathbb{T}_z^{i,i}W^{i,j} \leq \varepsilon_{W_x}\mathbb{T}_z^{i,i}, \quad \forall\, i, j \in 1, 2, \ldots, n$$

with superscripts i, j denoting matrix element (i, j). Since $\mathbb{T}_z \in \mathbb{D}_+^n$, then $\mathbb{T}_z^{i,i} > 0$, and so

$$-\varepsilon_{W_x} \leq \Psi^{i,j} - W^{i,j} \leq \varepsilon_{W_x},\ \forall i, j \in 1, 2, \ldots, n$$

giving the norm condition of (5). The remaining norm bounds of (5) can be obtained in a similar manner from (10b)–(10d). ■

Observe from (11) that the weights and biases of the robustified neural network, Ψ_z, Ψ_u, $\Psi_{g,z}$, $\Psi_{g,u}$ and biases β, β_g, can be extracted directly from the matrix variables of the problem: $\mathbb{T}_z$, $\mathbb{T}_g$, $\mathbb{T}_{u_1}$, $\mathbb{T}_{u_2}$, $\mathbb{Y}_z$, $\mathbb{Y}_u$, $\mathbb{Y}_{g,z}$, $\mathbb{Y}_{g,u}$, γ, $\gamma_{u,1}$ and $\gamma_{u,2}$. The conditions of the theorem are therefore linear in the decision variables, and so can be solved as a single SDP.

5. Numerical example: Robustifying model predictive control

To demonstrate the utility of Theorem 7 for a control-orientated application, consider the problem of approximating a model predictive control (MPC) policy to increase its robustness. The following example is based on that from Drummond et al. (2022a) [1].

Consider the discrete-time, time-invariant, linear control system

$$\begin{bmatrix} w_1[k+1] \\ w_2[k+1] \end{bmatrix} = \begin{bmatrix} 4/3 & -2/3 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} w_1[k] \\ w_2[k] \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v[k],$$

with state $w[k] \in \mathbb{R}^2$ and control action $v[k] \in \mathbb{R}$. Define the collection of future states and inputs as $\mathbf{w} = [w[k]^\top, w[k+1]^\top \ldots, w[k+N]^\top]$ and $\mathbf{v} = [v[k]^\top, v[k+1]^\top \ldots, v[k+N]^\top]$. The control action is to be obtained by solving the following quadratic program:

$$\begin{aligned} \text{minimize:} \quad & J(\mathbf{w}, \mathbf{v}) = w[k+N]^\top P w[k+N] \\ & + \sum_{i=1}^{N-1} w[k+i]^\top Q_i w[k+i] + v[k+i-1]^\top R_i v[k+i-1], \\ \text{subject to:} \quad & G\mathbf{v} \leq S_v w[k] + c, \end{aligned} \tag{13}$$

1. Code to generate these results can be found at: `https://github.com/r-drummond/convex-nn-synthesis-1norm`.

where $Q_i \in \mathbb{S}^2_{\succ 0}$, $P \in \mathbb{S}^2_{\succ 0}$, $R_i > 0 \; \forall i = 1, \ldots, N-1$, and the constraint set is $\mathbf{v} \in \mathcal{V}$. This quadratic program can be more compactly expressed as

$$\begin{aligned} \underset{\mathbf{v}}{\text{minimize}} \; & \mathbf{v}^\top H \mathbf{v} + 2w[k]^\top F^\top \mathbf{v}, \\ \text{subject to:} \quad & G\mathbf{v} \leq S_v w[k] + c. \end{aligned}$$

Here, we set the horizon length to $N = 10$ and the control action to be saturated at $-10 \leq \mathbf{v} \leq 10$, so $G = 0.1 \times \begin{bmatrix} I_m & -I_m \end{bmatrix}^\top$, $c = [1, \, 1, \, \ldots \, 1, \, 1]^\top$ and $S_v = 0$. Furthermore, the MPC quadratic cost function of (13) is parameterised by

$$P = \begin{bmatrix} 7.1667 & -4.2222 \\ -4.2222 & 4.6852 \end{bmatrix}, \; Q_i = \begin{bmatrix} 1 & -2/3 \\ -2/3 & 3/2 \end{bmatrix}, \; R_i = 1.$$

In a recent paper from Valmorbida and Hovd (2023), it was shown that the MPC control policy can be expressed as

$$x = (I_n - GH^{-1}G^\top)\phi(x) - (S_v + GH^{-1}F)w[k] - c, \tag{14a}$$

$$v[k] = f(w[k]) = -H^{-1}Fv[k] - H^{-1}G^\top \phi(x), \tag{14b}$$

with $\phi(x)$ being the ReLU function. By defining $\hat{x} = (I_n - GH^{-1}G^\top)^{-1}(x + Sw[k] + c) = \phi(x)$, it was shown in Drummond et al. (2024) that the MPC policy of (14) can be re-written as

$$\begin{aligned} \hat{x} &= \phi((I_n - GH^{-1}G^\top)\hat{x} - (S_v + GH^{-1}F)w[k] - c), \\ v[k] = g(w[k]) &= -H^{-1}Fw[k] - H^{-1}G^\top \phi(x) \\ &= -H^{-1}Fw[k] - H^{-1}G^\top \hat{x}\,, \end{aligned}$$

that is, as an implicit neural network in the form of (1) with weights and biases

$$\begin{aligned} &W_x = I_n - GH^{-1}G^\top, \; W_u = (S_v + GH^{-1}F), \; b = c, \\ &W_f = -H^{-1}G^\top, \; W_{f,u} = -H^{-1}F. \end{aligned}$$

Theorem 7 can then be directly applied to this MPC problem. A neural network of the form of (3) which trades off accuracy relative to the original MPC policy and its own robustness measured by Definition 2 can then be synthesized. In this example, the tolerances for the weights and biases are set to a fixed value of $\varepsilon_{W_x} = \varepsilon_{W_u} = \varepsilon_{W_{f,x}} = \varepsilon_{W_{f,u}} = \varepsilon$. Increasing ε opens up the search space for the weights, and so increases the neural network's robustness as it is that property which is optimised by Theorem 7. But, it also means that the gap between the weights of the original and robustified networks can be increased, which can reduce the accuracy relative to the original network of (1). Tuning this tolerance ε allows the trade-off between accuracy and robustness of (3) to be navigated.

Figures 1 and 2 compare the response of both the original MPC problem and the robustified neural network generated by Theorem 7. Figures 1 prioritises accuracy with respect to the MPC policy, with $\varepsilon = 10^{-5}$ whereas Figure 2 prioritises robustness, $\varepsilon = 10^{-1}$. This trade-off is captured by the responses, with the $\varepsilon = 10^{-5}$ response able to replicate that of the MPC (and capture the input saturation limits) unlike the $\varepsilon = 10^{-1}$ response.

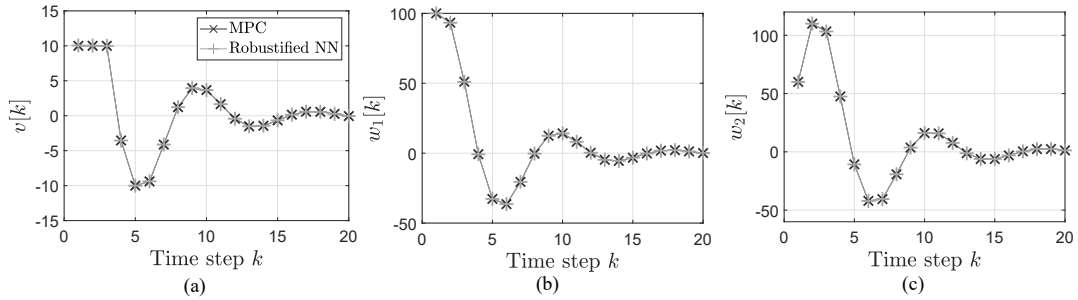

Figure 1: Comparison between MPC controller and neural network generated by Theorem 7 with a tolerance of $\varepsilon = 10^{-5}$. (a) Compares the inputs $v[k]$, (b) the state $w_1[k]$, and (c) the state $w_2[k]$.

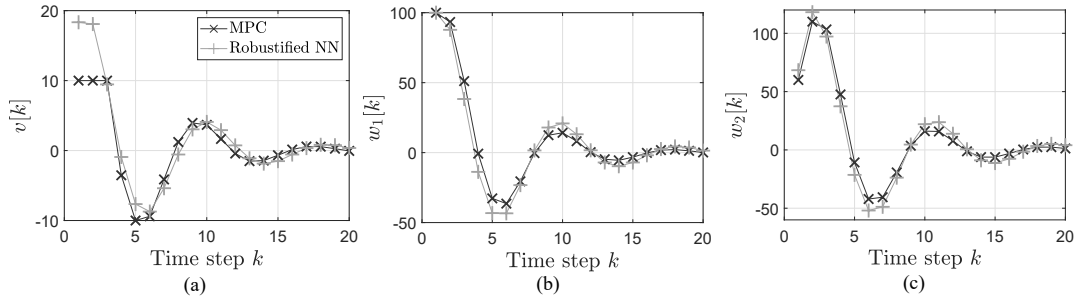

Figure 2: Comparison between MPC controller and neural network generated by Theorem 7 with a tolerance of $\varepsilon = 10^{-1}$. (a) Compares the inputs $v[k]$, (b) the state $w_1[k]$, and (c) the state $w_2[k]$.

A reduction in robustness is the price payed for this increased accuracy with respect to the original MPC policy. Figure 3 demonstrates this trade-off, which plots the robustified neural networks 1-norm bound against the weight tolerance ε. Using the definition of Definition 2, the gains $\gamma_{u,1} = \gamma_{u,2} = 0$ were set to have a single valuable capturing the bound for the 1-norm of the outputs. As ε increased, the bound γ reduced– indicating that a more robust network was generated. At $\varepsilon \approx 0.3$, the bound dropped suddenly because the solver had found the most robust network, which in this case was to map the states to zero. Even though this is not a particularly useful network, it is "optimally robust", a feature which emphasises the need to trade-off robustness against accuracy. Interestingly, even when ε was small and so could accurately capture the MPC response (see Figure 1), the synthesized network was more robust, with $\gamma = 1,751$ of the MPC reduced to $\gamma = 560$ for the neural network.

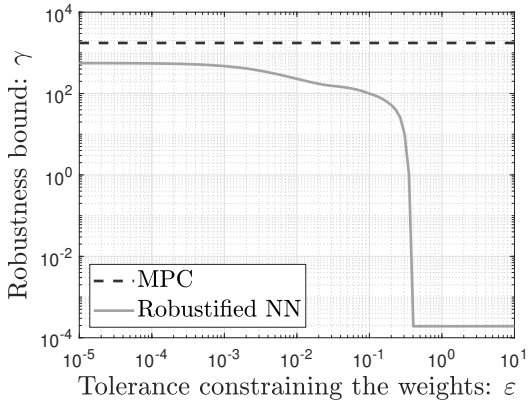

Figure 3: Trade-off between robustness (measured by γ of Definition 2) and accuracy (measured by ε of Definition 3) for the MPC problem.

Conclusions

A method to approximate a neural network by a more robust version was developed. By defining the robustness problem in terms of the 1-norm of the output, the proposed robust neural network synthesis method was shown to be a convex semi-definite programme. An application to robustifying model predictive control feedback policies demonstrated the applicability of the approach.

Appendix A.

We provide the matrices not given in the main text. The matrices of Lemmas 4, 5 and 6 are, respectively,

$$\Omega_z(\mathbb{T}_z) = \begin{bmatrix} 0_{2n_g\times 2n_g} & 0_{2n_g\times 2n_u} & 0_{2n_g\times n} & 0_{2n_g\times n_u} & 0_{2n_g} \\ \star & 0_{2n_u\times 2n_u} & 0_{2n_u\times n} & 0_{2n_u\times n_u} & 0_{2n_u} \\ \star & \star & He(\mathbb{T}_z\Psi_z - \mathbb{T}_z) & \mathbb{T}_z\Psi_u & 0_n \\ \star & \star & \star & 0_{n_u\times n_u} & 0_{n_u} \\ \star & \star & \star & \star & 0 \end{bmatrix}, \tag{16}$$

$$\Omega_u(\mathbb{T}_{u,1},\mathbb{T}_{u,2}) = \begin{bmatrix} 0_{2n_g\times 2n_g} & 0_{2n_g\times 2n_u} & 0_{2n_g\times n} & 0_{2n_g\times n_u} & 0_{2n_g} \\ \star & -\frac{1}{2}\mathbb{T}_{u,2}\begin{bmatrix} I_{n_u} & 0_{n_u\times n_u} \\ \star & I_{n_u}\end{bmatrix} & 0_{2n_u\times n} & 0_{2n_u\times n_u} & -\frac{1}{2}\mathbb{T}_{u,1}\begin{bmatrix}\mathbf{1}_{n_u}\\ \mathbf{1}_{n_u}\end{bmatrix} \\ \star & \star & 0_{n\times n} & 0_{n\times n_u} & 0_n \\ \star & \star & \star & -\frac{1}{2}\mathbb{T}_{u,2}I_{n_u} & 0_{n_u} \\ \star & \star & \star & \star & \mathbb{T}_{u,1}\varepsilon_{u,1} + \varepsilon_{u,2}\mathbb{T}_{u,2} \end{bmatrix}, \tag{17}$$

$$\Omega_g(\mathbb{T}_g) = \begin{bmatrix} \begin{bmatrix} -\mathbb{T}_g & 0_{n_g\times n_g} \\ \star & -\mathbb{T}_g\end{bmatrix} & 0_{2n_g\times 2n_u} & \begin{bmatrix}\mathbb{T}_g\Psi_{g,z}\\ -\mathbb{T}_g\Psi_{g,z}\end{bmatrix} & \begin{bmatrix}\mathbb{T}_g\Psi_{g,u}\\ -\mathbb{T}_g\Psi_{g,u}\end{bmatrix} & 0_{2n_g} \\ \star & 0_{2n_u\times 2n_u} & 0_{2n_u\times n} & 0_{2n_u\times n_u} & 0_{2n_u} \\ \star & \star & 0_{n\times n} & 0_{n\times n_u} & 0_n \\ \star & \star & \star & 0_{n_u\times n_u} & 0_{n_u} \\ \star & \star & \star & \star & 0 \end{bmatrix}. \tag{18}$$

Finally, the following matrices are used in Theorem 7.

$$\check{\Omega}_z(\mathbb{T}_z,\mathbb{Y}_z,\mathbb{Y}_u) = \begin{bmatrix} 0_{2n_g\times 2n_g} & 0_{2n_g\times 2n_u} & 0_{2n_g\times n} & 0_{2n_g\times n_u} & 0_{2n_g} \\ \star & 0_{2n_u\times 2n_u} & 0_{2n_u\times n} & 0_{2n_u\times n_u} & 0_{2n_u} \\ \star & \star & He(\mathbb{Y}_z - \mathbb{T}_z) & \mathbb{Y}_u & 0_n \\ \star & \star & \star & 0_{n_u\times n_u} & 0_{n_u} \\ \star & \star & \star & \star & 0 \end{bmatrix}, \tag{19}$$

$$\check{\Omega}_g(\mathbb{T}_g,\mathbb{Y}_{g,z},\mathbb{Y}_{g,u}) = \begin{bmatrix} \begin{bmatrix} -\mathbb{T}_g & 0_{n_g\times n_g} \\ \star & -\mathbb{T}_g\end{bmatrix} & 0_{2n_g\times 2n_u} & \begin{bmatrix}\mathbb{Y}_{g,z}\\ -\mathbb{Y}_{g,z}\end{bmatrix} & \begin{bmatrix}\mathbb{Y}_{g,u}\\ -\mathbb{Y}_{g,u}\end{bmatrix} & 0_{2n_g} \\ \star & 0_{2n_u\times 2n_u} & 0_{2n_u\times n} & 0_{2n_u\times n_u} & 0_{2n_u} \\ \star & \star & 0_{n\times n} & 0_{n\times n_u} & 0_n \\ \star & \star & \star & 0_{n_u\times n_u} & 0_{n_u} \\ \star & \star & \star & \star & 0 \end{bmatrix}, \tag{20}$$

$$\Omega_\gamma(\gamma,\gamma_{u,1},\gamma_{u,2}) = \begin{bmatrix} 0_{2n_g\times 2n_g} & 0_{2n_g\times 2n_u} & 0_{2n_g\times n} & 0_{2n_g\times n_u} & \frac{1}{2}\begin{bmatrix}\mathbf{1}_{n_g}\\ \mathbf{1}_{n_g}\end{bmatrix} \\ \star & -\frac{1}{2}\gamma_{u,2}\, I_{2n_u} & 0_{2n_u\times n} & 0_{2n_u\times n_u} & -\frac{\gamma_{u,1}}{2}\begin{bmatrix}\mathbf{1}_{n_u}\\ \mathbf{1}_{n_u}\end{bmatrix} \\ \star & \star & 0_{n\times n} & 0_{n\times n_u} & 0_n \\ \star & \star & \star & -\frac{1}{2}\gamma_{u,2}I_{n_u} & 0_{n_u} \\ \star & \star & \star & \star & -\gamma \end{bmatrix}. \tag{21}$$

Acknowledgments

Ross Drummond was supported by a UK Intelligence Community Research Fellowship from the Royal Academy of Engineering.

References

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

Nikita E Barabanov and Danil V Prokhorov. Stability analysis of discrete-time recurrent neural networks. *IEEE Transactions on Neural Networks*, 13(2):292–303, 2002.

Yun-Chung Chu and Keith Glover. Bounds of the induced norm and model reduction errors for systems with repeated scalar nonlinearities. *IEEE Transactions on Automatic Control*, 44(3): 471–483, 1999a.

Yun-Chung Chu and Keith Glover. Stabilization and performance synthesis for systems with repeated scalar nonlinearities. *IEEE Transactions on Automatic Control*, 44(3):484–496, 1999b.

Ross Drummond, Stephen Duncan, Mathew Turner, Patricia Pauli, and Frank Allgower. Bounding the difference between model predictive control and neural networks. In *Learning for Dynamics and Control Conference*, pages 817–829. PMLR, 2022a.

Ross Drummond, Matthew C Turner, and Stephen R Duncan. Reduced-order neural network synthesis with robustness guarantees. *IEEE Transactions on Neural Networks and Learning Systems*, 2022b.

Ross Drummond, Ross Baldivieso, and Giorgio Valmorbida. Mapping back and forth between model predictive control and neural networks. In *Submitted to Learning for Dynamics and Control*. PMLR, 2024.

Geir E Dullerud and Fernando Paganini. *A course in robust control theory: A convex approach*, volume 36. Springer Science & Business Media, 2013.

Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of Lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Luca Furieri, Clara Lucía Galimberti, and Giancarlo Ferrari-Trecate. Neural system level synthesis: Learning over all stabilizing policies for nonlinear systems. In *Procs. of the Conference on Decision and Control (CDC)*, pages 2765–2770. IEEE, 2022.

Michael Green and David JN Limebeer. *Linear robust control*. Courier Corporation, 2012.

Neelay Junnarkar, He Yin, Fangda Gu, Murat Arcak, and Peter Seiler. Synthesis of stabilizing recurrent equilibrium network controllers. In *Procs. of the Conference on Decision and Control (CDC)*, pages 7449–7454. IEEE, 2022.

Jiaqi Li, Ross Drummond, and Stephen R Duncan. Robust error bounds for quantised and pruned neural networks. In *Learning for Dynamics and Control*, pages 361–372. PMLR, 2021.

Matthew Newton and Antonis Papachristodoulou. Stability of non-linear neural feedback loops using sum of squares. In *Prcos. of the Conference on Decision and Control (CDC)*, pages 6000–6005. IEEE, 2022.

Matthew Newton and Antonis Papachristodoulou. Sparse polynomial optimisation for neural network verification. *Automatica*, 157:111233, 2023.

Patricia Pauli, Dennis Gramlich, Julian Berberich, and Frank Allgöwer. Linear systems with neural network nonlinearities: Improved stability analysis via acausal Zames-Falb multipliers. In *Procs. of the Conference on Decision and Control (CDC)*, pages 3611–3618. IEEE, 2021a.

Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgöwer. Training robust neural networks using Lipschitz bounds. *IEEE Control Systems Letters*, 6:121–126, 2021b.

Ashwini Pokle, Zhengyang Geng, and J Zico Kolter. Deep equilibrium approaches to diffusion models. *Advances in Neural Information Processing Systems*, 35:37975–37990, 2022.

Max Revay, Ruigang Wang, and Ian R Manchester. Recurrent equilibrium networks: Flexible dynamic models with guaranteed stability and robustness. *IEEE Transactions on Automatic Control*, 2023.

Carl R Richardson, Matthew C Turner, and Steve R Gunn. Strengthened Circle and Popov criteria for the stability analysis of feedback systems with ReLU neural networks. *IEEE Control Systems Letters*, 2023.

Gunter Stein. Respect the unstable. *IEEE Control systems magazine*, 23(4):12–25, 2003.

Giorgio Valmorbida and Morten Hovd. Quadratic programming with ramp functions and fast online QP-MPC solutions. *Automatica*, 153:111011, 2023.

Ruigang Wang and Ian Manchester. Direct parameterization of Lipschitz-bounded deep networks. In *International Conference on Machine Learning*, pages 36093–36110. PMLR, 2023.

Zi Wang, Gautam Prakriya, and Somesh Jha. A quantitative geometric approach to neural-network smoothness. *Advances in Neural Information Processing Systems*, 35:34201–34215, 2022.

Zi Wang, Bin Hu, Aaron J Havens, Alexandre Araujo, Yang Zheng, Yudong Chen, and Somesh Jha. On the scalability and memory efficiency of semidefinite programs for Lipschitz constant estimation of neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

George Zames and PL Falb. Stability conditions for systems with monotone and slope-restricted nonlinearities. *SIAM Journal on Control*, 6(1):89–108, 1968.