
Article:
Drummond, R. orcid.org/0000-0002-2586-1718, Turner, M.C. orcid.org/0000-0003-2161-7635 and Duncan, S.R. orcid.org/0000-0002-9525-7305 (2024) Reduced-order neural network synthesis with robustness guarantees. IEEE Transactions on Neural Networks and Learning Systems, 35 (1). pp. 1182-1191. ISSN 2162-237X

https://doi.org/10.1109/tnnls.2022.3182893

Reduced-Order Neural Network Synthesis with Robustness Guarantees

Ross Drummond, Matthew C. Turner and Stephen R. Duncan

***Abstract*—In the wake of the explosive growth in smartphones and cyberphysical systems, there has been an accelerating shift in how data is generated away from centralised data towards on-device generated data. In response, machine learning algorithms are being adapted to run locally on board, potentially hardware limited, devices to improve user privacy, reduce latency and be more energy efficient. However, our understanding of how these device-orientated algorithms behave and should be trained is still fairly limited. To address this issue, a method to automatically synthesize *reduced-order* neural networks (having fewer neurons) approximating the input/output mapping of a larger one is introduced. The reduced-order neural network's weights and biases are generated from a convex semi-definite programme that minimises the worst-case approximation error with respect to the larger network. Worst-case bounds for this approximation error are obtained and the approach can be applied to a wide variety of neural networks architectures. What differentiates the proposed approach to existing methods for generating small neural networks, e.g. pruning, is the inclusion of the worst-case approximation error directly within the training cost function, which should add robustness to out-of-sample data-points. Numerical examples highlight the potential of the proposed approach. The overriding goal of this paper is to generalise recent results in the robustness analysis of neural networks to a robust synthesis problem for their weights and biases.**

I. Introduction

As smartphones get increasingly integrated into our daily lives and the numbers of both cyberphysical systems and smart devices continues to grow, there has been a noticeable evolution in the way many large data sets are being generated. In fact, Cisco [13] predicted that in 2021, whilst 20.6 ZB of data (e.g. large ecommerce site records) will be handled by cloud-based approaches in large data-centres, this amount will be dwarfed by the 850 ZB generated by local devices [40]. In response to data sources becoming more device-centric, there has been a shift in focus for many machine learning algorithms towards both implementing and training them locally on (potentially hardware limited) devices. Running the algorithms on the devices represents a radical shift away from traditional *centralised learning* where the data and algorithms are stored and processed in the cloud but, as described in [40], brings the benefits of i) increased user privacy as the data is not transmitted to a centralised server ii) reduced latency since the algorithms can react immediately to newly generated data from the device and iii) improved energy efficiency mostly because the data and algorithm outputs do not have to be constantly transferred to and from the cloud. However, running algorithms locally on devices brings its own issues, most notably in dealing with the devices' limited computational power, memory and energy storage. Overcoming these hardware constraints has motivated substantial efforts on improving algorithm design, particularly towards developing leaner, more efficient neural networks [36].

Two popular approaches to make neural network algorithms leaner and more hardware-conscious are i) quantised neural networks [34], [7], [35], where fixed-point arithmetic is used to accelerate the computational speed and reduce memory footprint, and ii) pruned neural networks [27], [4], [18], [19], [32], [31], [22], [17], [12], [23], [26], [15], [30], [24], where, typically, the weights contributing least to the function mapping are removed, promoting sparsity in the weights. Both of these approaches have achieved impressive results. For instance, by quantising, [25] was able to reduce model size by $> 20\%$ without any noticeable loss in accuracy when evaluated on the CIFAR-10 benchmark and [16] demonstrated that between 50-80% of its model weights could be pruned with little impact on performance [36]. However, our understanding of neural network reduction methods such as these remains lacking and reliably predicting their performance remains a challenge. Illustrating this point, [27] stated that for pruned neural networks "our results suggest the need for more careful baseline evaluations in future research on structured pruning methods" with a similar sentiment raised in [4]: "our clearest finding is that the community suffers from a lack of standardized benchmarks and metrics". These quotes indicate a need for robust evaluation methods for lean neural network designs, a perspective explored in this work.

Contribution: This paper introduces a method to automatically synthesize neural networks of reduced dimensions (meaning fewer neurons) from a trained larger one, as illustrated in Figure 1. These smaller networks are termed *reduced-order neural networks* since the approach was inspired by reduced order modelling in control theory [14]. The weights and biases of the reduced order network are generated from the solution of a semi-definite program (SDP)- a class of well-studied convex problems [5] combining a linear cost function with a linear matrix

Ross Drummond and Stephen R. Duncan are with the Department of Engineering Science, University of Oxford, OX1 3PJ, Oxford, United Kingdom. Email: {ross.drummond,stephen.duncan}@eng.ox.ac.uk.

Mathew C. Turner is with the Department of Electronics and Computer Science, University of Southampton, Southampton, UK, SO17 1BJ. Email: m.c.turner@soton.ac.uk

inequality (LMI) constraint- which minimises the worst-case approximation error with respect to the larger network. Bounds are also obtained for this worst-case approximation error and so the performance of the network reduction is guaranteed. In this way, the method is said to be "robust" as it ensures the approximation error of the reduced-order neural network remains bounded for all input data in certain pre-defined sets, in a manner specified by the bound of Theorem 1.

What separates the proposed synthesis approach to the existing methods for generating efficient neural networks, e.g. pruning, is the inclusion of the worst-case approximation error of the reduced-order neural network directly within the cost function for computing the weights and biases. It is expected that this approach should offer two main advantages over classical pruning methods:

1) The method is robust in the sense that it provides guarantees of the approximation error with respect to the full-order network, unlike with pruning.
2) The method is one of the first to do automatic neural network *synthesis* from the solution of a robust optimisation problem, with the weights and biases of the reduced-order neural networks generated in one-shot by solving a convex semi-definite program. Besides being of theoretical interest as an alternative to training via backpropogation, the main advantage of this approach is that it allows the worst-case error to be included directly within the training cost function which may result in out-of-sample generality in worst-case settings.

Whilst the presented results are still preliminary, their focus on robust neural network synthesis introduces a new set of of tools to generate lean neural networks which should have more reliable out-of-sample performance, and which are equipped with approximation error bounds. The broader goals of this work are to translate recent results on the verification of NN robustness using SDPs [11], [33] into a synthesis problem, mimicking the progression from absolute stability theory [39] to robust control synthesis [9] witnessed in control theory during the 1980s. In this way, this work carries on the tradition of control theorists exploring the connections between robust control theory and neural networks, as witnessed since the 1990s with Glover [6], Barabanov [3], Angeli [2] and Narendra [21].

A. Notation

Non-negative real vectors of dimension n are denoted $\mathbb{R}^n_+$. A positive (negative) definite matrix Ω is denoted $\Omega \succ (\prec)\ 0$. Non-negative diagonal matrices of dimension $n \times n$ are $\mathbb{D}^n_+$. The matrix of zeros of dimension $n \times m$ is $\mathbf{0}_{n\times m}$ and the vector of zeros of dimension n is $\mathbf{0}_n$. The identity matrix of size n is I_n. The vector of 1s of dimension n is $\mathbf{1}_n$ and the $n\times m$ matrix of 1s is $\mathbf{1}_{n\times m}$. The i^{th} element of a vector x is denoted x_i unless otherwise defined in the text. The $\star$ notation is adopted to represent symmetric matrices in a compact form, e.g.

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} = \begin{bmatrix} A & B \\ \star & C \end{bmatrix}. \tag{1}$$

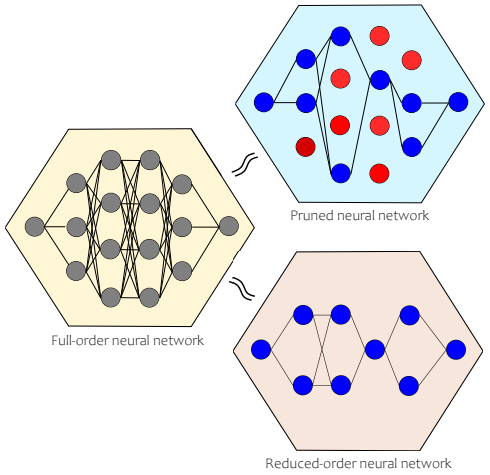


Fig. 1: Illustration of two different approximations of a neural network (termed the full-order network) to enable it to be run on limited hardware. One approach is to use network pruning to make the weights sparse while the second is to develop a reduced-order network with fewer neurons. This paper proposes a method to synthesize the weights and biases of the reduced order network such that they robustly minimise the approximation error with respect to the full order network.

B. Neural networks

The neural networks considered will be treated as functions $f(x) : \mathcal{X} \to \mathcal{F}$ mapping input vectors of size $x \in \mathbb{R}^{n_x}$ to output vectors of dimension $f(x) \in \mathbb{R}^{n_f}$. In a slight abuse of notation, $\phi(\cdot) : \mathbb{R}^{n_k} \to \mathbb{R}^{n_k}$ will refer to mappings of both scalars and vectors, with the vector operation applied element-wise. The full-order neural network will be composed of l hidden layers, with the k^{th} layer being composed of n_k neurons. The total number of neurons in the full-order neural network is $N = \sum_{k=1}^{l} n_k$. Similarly, the reduced-order neural network will be composed of λ hidden layers with the k^{th} layer being composed of m_k neurons. The total number of neurons in the reduced-order network is $M = \sum_{k=1}^{\lambda} m_k$. The dimension of the domain of the activation functions is defined as $\bar{N} = N - n_l + n_x$ (full-order network) and $\bar{M} = M - n_\lambda + n_x$ (reduced-order network).

II. Problem statement

In this section, the general problem of synthesizing reduced-order NNs is posed. Consider a nonlinear function $f(x) : \mathcal{X} \to \mathcal{F}$ mapping input data $x \in \mathcal{X}$ to an output set $\mathcal{F}$. The goal of this work is to generate a "*simpler*" function $g(x) : \mathcal{X} \to \mathcal{G}$ that is as "*close*" as possible to $f(x)$ for all $x \in \mathcal{X}$. Here, "*simpler*" will refer to the dimension of the reduced-order neural network's weight matrix being smaller than the full-order one and

"*close*"-ness relates to the approximation error between the two functions $f(x)$ and $g(x)$ measured by the induced 2-norm $\sup_{x\in\mathcal{X}} \|f(x)-g(x)\|_2$. The goal is to automatically synthesize the simpler functions $g(x)$ from the solution of a convex problem and obtain worst-case bounds for approximation error with respect to the larger neural network $f(x)$ for all $x \in \mathcal{X}$.

To ensure that the function approximation problem remains feasible, structure is added to the set $\mathcal{F}$. It is assumed that the function being approximated $f(x)$ is generated by a feed-forward neural network

$$x^0 = x, \tag{2a}$$
$$x^{k+1} = \phi(W^k x^k + b^k), \quad k = 0, 1, \ldots, l-1, \tag{2b}$$
$$f(x) = W^l x^l + b^l. \tag{2c}$$

Here, the input data $x^0 = x \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ is mapped through the nonlinear activation functions $\phi(\cdot)$ (which could be the standard choices of ReLU, sigmoid, tanh or any function that satisfies a quadratic constraint as given in Section III-B) element-wise with the weight matrices $W^0 \in \mathbb{R}^{n_1 \times n_x}$, $W^k \in \mathbb{R}^{n_{k+1} \times n_k}$ and biases $b^k \in \mathbb{R}^{n_{k+1}}$, $k = 0, \ldots, l-1$. Whilst the results are described for feed-forward neural networks, the method can be generalised to other network architectures, such as recurrent and even implicit neural networks [10]. As an aside, verifying the well-posedness of implicit neural networks has a strong connection to that of Lurie systems with feed-through terms [37].

The network (2) can be equivalently written in the implicit form

$$\check{x} = \phi(W\check{x} + W_0 x + b), \qquad \phi(.): \mathbb{R}^N \mapsto \mathbb{R}^N, \tag{3a}$$
$$f(x) = W^f \check{x} + b^l, \tag{3b}$$

where

$$\check{x} = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^l \end{bmatrix}, W = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ W^1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & 0 \\ 0 & \ldots & W^{l-1} & 0 \end{bmatrix}, W_0 = \begin{bmatrix} W^0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{4a}$$

$$b = \begin{bmatrix} b^0 \\ b^1 \\ \vdots \\ b^{l-1} \end{bmatrix}, \quad W^f = \begin{bmatrix} 0, & \ldots, & 0, & W^l \end{bmatrix}. \tag{4b}$$

The neural network $f(x)$ (which will be referred to as the *full-order neural network*) is to be approximated by another neural network $g(x) \in \mathcal{G}$ (referred to as the *reduced-order neural network*) of a smaller dimension

$$z^0 = x, \tag{5a}$$
$$z^{k+1} = \phi\left(\sum_{i=0}^{\lambda-1} \Psi^{k+1,i} z^i + \beta^k\right), \quad k = 0, 1, \ldots, \lambda-1, \tag{5b}$$
$$g(x) = \Psi^\lambda z^\lambda + \beta^\lambda. \tag{5c}$$

The weights and biases in this neural network are $\Psi^{k,i} \in \mathbb{R}^{m_{k+1} \times m_i}$, $\beta^k \in \mathbb{R}^{m_{k+1}}$, $k = 0, \ldots, \lambda-1$. The network structure in (5b) is general, and allows for implicitly defined networks [10]. This generality follows from the lack of structure imposed on the matrices used in the synthesis procedure. However, by adding structure, the search can be limited to, for example, feed-forward networks, which are simpler to implement.

Similar to the full-order case, the network (5) can be written as

$$\check{z} = \phi(\Psi\check{z} + \Psi_0 x + \beta), \quad \phi(\cdot): \mathbb{R}^M \mapsto \mathbb{R}^M, \tag{6a}$$
$$g = \Psi^f \check{z} + \beta^\lambda, \tag{6b}$$

where

$$\check{z} = \begin{bmatrix} z^1 \\ z^2 \\ \vdots \\ z^\lambda \end{bmatrix}, \Psi = \begin{bmatrix} \Psi^{1,1} & \Psi^{1,2} & \ldots & \Psi^{1,\lambda} \\ \Psi^{2,1} & \Psi^{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \Psi^{\lambda-1,\lambda} \\ \Psi^{\lambda,1} & \ldots & \Psi^{\lambda,\lambda-1} & \Psi^{\lambda,\lambda} \end{bmatrix}, \tag{7a}$$

$$\Psi_0 = \begin{bmatrix} \Psi^{1,0} \\ \Psi^{2,0} \\ \vdots \\ \Psi^{\lambda,0} \end{bmatrix}, \beta = \begin{bmatrix} \beta^0 \\ \beta^1 \\ \vdots \\ \beta^{\lambda-1} \end{bmatrix}, \Psi^f = \begin{bmatrix} 0, & \ldots, & 0, & \Psi^\lambda \end{bmatrix}. \tag{7b}$$

In this work, the dimension of the reduced-order network is fixed and the problem is to find the reduced-order NN's parameters, being the weights $\Psi^{k,i}$ and biases β^k, that minimise the worst-case approximation error between the full and reduced order neural networks for all $x \in \mathcal{X}$. In practice, the dimension of the reduced-order network should be fixed to the minimum value for which Proposition 1 can be solved to give a sufficient level of performance, as typically increasing the dimension of this neural network should lead to improved approximations to the full-order one, as larger networks will be more expressive allowing them to more accurately approximate highly nonlinear functions. The main tool used for this reduced-order NN synthesis problem is the outer approximation of the NN's input set $\mathcal{X}$, nonlinear activation function's gains $\phi(\cdot)$ and the output error $f(x)-g(x)$ by quadratic constraints. These outer approximations enable the robust weight synthesis problem to be stated as a convex SDP, albeit at the expense of introducing conservatism into the analysis.

III. Quadratic Constraints

In this section, the quadratic constraints for the convex outer approximations of the various sets of interest of the reduced NN synthesis problem are defined. These characterisations are posed in the framework of [11], which in turn was inspired by the integral quadratic constraint framework of [29] and the classical absolute stability results for Lurie systems [20].

A. Quadratic constraint: Input set

The input data $x \in \mathcal{X}$ is restricted to the hyper-rectangle $\mathcal{X}_\infty$.

Definition 1: Define the hyper-rectangle $\mathcal{X}_\infty = \{x : \underline{x}_i \leq x_i \leq \overline{x}_i, i = 1, \ldots, n_x\}$. If $x \in \mathcal{X}_\infty$ then $[x^T \ 1] P_{\mathcal{X}_\infty} [x^T \ 1]^T \geq 0$ where

$$P_{\mathcal{X}_\infty} = \begin{bmatrix} -\tau_{x_\infty} & \frac{\tau_{x_\infty}}{2}(\underline{x} + \overline{x}) \\ \star & -\underline{x}^T \tau_{x_\infty} \overline{x} \end{bmatrix}, \quad \tau_{x_\infty} \in \mathbb{D}_+^{n_x}. \tag{8}$$

Note that the input set constraint characterised by Definition 1 can be equivalently written as

$$\omega(x)^T \Pi_\infty \omega(x) \geq 0 \tag{9}$$

where

$$\Pi_\infty = \begin{bmatrix} -\tau_{x_\infty} & 0_{n_x \times N} & 0_{n_x \times M} & \frac{\tau_{x_\infty}}{2}(\underline{x} + \overline{x}) \\ \star & 0_{N \times N} & 0_{N \times M} & 0_{N \times 1} \\ \star & \star & 0_{M \times M} & 0_{M \times 1} \\ \star & \star & \star & -\underline{x}^T \tau_{x_\infty} \overline{x} \end{bmatrix}$$

and

$$\mu(x) = \begin{bmatrix} x^T & \check{x}^T & \check{z}^T & 1 \end{bmatrix}^T. \tag{10}$$

B. Quadratic constraint: Activation functions

The main obstacle to any robustness-type result for neural networks is accounting for the nonlinear activation functions $\phi(\cdot)$. To address this issue, the following function properties are introduced.

Definition 2: The activation function $\phi(s) : \mathcal{S} \subset \mathbb{R} \to \mathbb{R}$ satisfying $\phi(0) = 0$ is said to be *sector bounded* if

$$\frac{\phi(s)}{s} \in [0, \delta] \quad \forall s \in \mathcal{S}, \quad \delta > 0, \tag{11a}$$

and *slope restricted* if

$$\frac{\phi(s_1) - \phi(s_2)}{s_1 - s_2} \in [\underline{\beta}, \beta], \quad \forall s_1, s_2 \neq s_1 \in \mathcal{S}, \quad \beta > 0. \tag{11b}$$

If $\underline{\beta} = 0$ then the nonlinearity is *monotonic* and if $\phi(0) = 0$ then the slope restriction implies sector boundedness. The activation function $\phi(s)$ is *bounded* if

$$\phi(s) \in [\underline{c}, \overline{c}], \quad \forall s \in \mathcal{S}, \tag{11c}$$

it is *positive* if

$$\phi(s) \geq 0, \quad \forall s \in \mathcal{S}, \tag{11d}$$

its *complement is positive* if

$$\phi(s) - s \geq 0, \quad \forall s \in \mathcal{S}, \tag{11e}$$

and it satisfies the *complementarity condition* if

$$(\phi(s) - s)\phi(s) = 0, \quad \forall s \in \mathcal{S}. \tag{11f}$$

Most popular activation functions, including the ReLU, (shifted-)sigmoid and tanh satisfy one or more of these conditions, as illustrated in Table I. As the number of properties satisfied by $\phi(\cdot)$ increases, the characterisation of this function within the robustness analysis improves, often resulting in less conservative results. It is also noted that to satisfy $\phi(0) = 0$ some activation functions may require a shift, e.g. the sigmoid, or they may require transformations to satisfy additional function properties, as demonstrated in the representation of the LeakyReLU as a ReLU + linear term function.

As is well-known from control theory [20], functions with these specific properties are important for robustness analysis problems because they can be characterised by quadratic constraints.

Lemma 1: Consider the vectors $y, y_1 \in \mathbb{R}^{n_y}$, and $\upsilon \in \mathbb{R}^{n_\upsilon}$ that are mapped component-wise through the activation functions $\phi(\cdot) : \mathbb{R}^{n_y} \to \mathbb{R}^{n_y}$ and $\tilde{\phi}(\cdot) : \mathbb{R}^{n_\upsilon} \to \mathbb{R}^{n_\upsilon}$. If $\phi(y)$ is *sector-bounded*, then

$$(\delta y - \phi(y))^T \mathbf{T}^{\mathrm{s}} \phi(y) \geq 0, \quad \forall y \in \mathbb{R}^{n_y}, \ \mathbf{T}^{\mathrm{s}} \in \mathbb{D}_+^{n_y}; \tag{12a}$$

slope-restricted then

$$(\beta(y - y_1) - (\phi(y) - \phi(y_1))^T \mathbf{T}^{\mathrm{sl}} (\phi(y) - \phi(y_1) - \underline{\beta}(y - y_1)) \geq 0; \quad \forall \{y, y_1\} \in \mathbb{R}^{n_y}, \ \mathbf{T}^{\mathrm{sl}} \in \mathbb{D}_+^{n_y}; \tag{12b}$$

bounded then

$$(\overline{c} - \phi(y))^T \mathbf{T}^B (\phi(y) - \underline{c}) \geq 0, \quad \forall y \in \mathbb{R}^{n_y}, \ \mathbf{T}^B \in \mathbb{D}_+^{n_y}; \tag{12c}$$

positive then

$$(\mathbf{T}^+)^T \phi(y) \geq 0, \quad \forall y \in \mathbb{R}^{n_y}, \ \mathbf{T}^+ \in \mathbb{R}_+^{n_y}; \tag{12d}$$

such that is *complement is positive* then

$$(\mathbf{T}^{c+})^T (\phi(y) - y) \geq 0, \quad \forall y \in \mathbb{R}^{n_y}, \ \mathbf{T}^{c+} \in \mathbb{R}_+^{n_y}. \tag{12e}$$

If $\phi(y)$ satisfies the *complementary* condition then

$$(\phi(y) - y)^T \mathbf{T}^0 \phi(y) = 0, \quad \forall y \in \mathbb{R}^{n_y}, \ \mathbf{T}^0 \in \mathbb{D}^{n_y}. \tag{12f}$$

Additionally, if both $\phi(y)$ and $\tilde{\phi}(\upsilon)$ and their complements are positive then so are the *cross terms*

$$\tilde{\phi}(\upsilon)^T \mathbf{T}^{\times} (\phi(y) - y) \geq 0, \ \forall \upsilon \in \mathbb{R}^{n_\upsilon}, y \in \mathbb{R}^{n_y}, \ \mathbf{T}^{\times} \in \mathbb{R}_+^{n_\upsilon \times n_y}, \tag{12g}$$

$$\tilde{\phi}(\upsilon)^T \mathbf{T}^{\otimes} \phi(y) \geq 0, \ \forall \upsilon \in \mathbb{R}^{n_\upsilon}, y \in \mathbb{R}^{n_y}, \ \mathbf{T}^{\otimes} \in \mathbb{R}_+^{n_\upsilon \times n_y}. \tag{12h}$$

Inequalities (12a)-(12f) are well-known however the cross terms (12g)-(12h) acting jointly on activation function pairs are less so.

Remark 1: Lemma 1 is established globally, that is for all $y \in \mathbb{R}^{n_y}$. Some activation functions $\phi(\cdot)$ may be defined locally, or the sector, slope bounds may be tighter for restricted values of their arguments. In such cases, local versions of Lemma 1 may give less conservative results. $\star$

The characterisation of the nonlinear activation functions via quadratic constraints allows the neural network robustness analysis to be posed as a SDP- with the various λ's in Lemma 1 being decision variables. Such an approach has been used in [11], [14], [3], and elsewhere, for neural networks robustness problems, with the conservatism of this approach coming from the obtained worst-case bounds holding for all nonlinearities satisfying the quadratic constraints. In this work, the aim is to extend this quadratic constraint framework for neural network robustness analysis problems to a synthesis problem.

$\phi(\cdot)$ property	Shifted sigmoid	tanh	ReLU	ELU
Sector bounded	✓	✓	✓	✓
Slope restricted	✓	✓	✓	✓
Bounded	✓	✓	×	×
Positive	×	×	✓	×
Positive complement	×	×	✓	×
Complementarity condition	×	×	✓	×

TABLE I: Properties of commonly used activation functions, including the sigmoid, tanh, rectified linear unit ReLU and exponential linear unit (ELU). The properties of other functions, such as the LeakyReLU, can also be inferred.

A quadratic constraint characterisation of both the reduced and full-order neural networks can then be written, with the following lemma being the application of Lemma 1 for both the reduced and full-order neural networks.

Lemma 2: If the activation function $\phi(\cdot)$ satisfies one or more of the quadratic constraints of Lemma 1, then there exists a matrix

$$\Lambda = \begin{bmatrix} \mathbf{0}_{n_x \times n_x} & \Lambda_{12} & \Lambda_{13} & \Lambda_{14} \\ \star & \Lambda_{22} & \Lambda_{23} & \Lambda_{24} \\ \star & \star & \Lambda_{33} & \Lambda_{34} \\ \star & \star & \star & \Lambda_{44} \end{bmatrix}, \tag{13}$$

defined by the $\mathbf{T}^i$'s ($i \in \{\mathrm{s}, \mathrm{sl}, +, c+, B, 0, \times, \otimes\}$) of Lemma 1 such that

$$\mu(x)^T \Lambda \mu(x) \geq 0, \quad \forall x \in \mathcal{X}. \tag{14}$$

Proof. The construction of Λ for the sector nonlinearity associated with the full-order network is shown. Lemma 1 implies that for a matrix $\mathbf{T}^{\mathrm{s}} \in \mathbb{D}_+^N$

$$2\phi(\xi)^T \mathbf{T}^{\mathrm{s}}(\xi - \phi(\xi)) \geq 0$$

where, from equation (3), $\xi = W\check{x} + W_0 x + b$. Noting that $\phi(\xi) = \check{x}$, expanding the above becomes

$$2\check{x}^T \mathbf{T}^{\mathrm{s}}(W\check{x} + W_0 x + b - \check{x}) \geq 0$$

and majorising it gives

$$\begin{bmatrix} x \\ \check{x} \\ \check{z} \\ 1 \end{bmatrix}^T \begin{bmatrix} 0 & W_0^T \mathbf{T}^{\mathrm{s}} & 0 & 0 \\ \star & -2\mathbf{T}^{\mathrm{s}} + \mathbf{T}^{\mathrm{s}} W + W^T \mathbf{T}^{\mathrm{s}} & 0 & \mathbf{T}^{\mathrm{s}} b \\ \star & \star & 0 & 0 \\ \star & \star & \star & 0 \end{bmatrix} \begin{bmatrix} x \\ \check{x} \\ \check{z} \\ 1 \end{bmatrix} \geq 0. \tag{15}$$

This clearly takes the form of the inequality in (14). All other cases are derived similarly. Appendix 1 details the characterisation of Λ for the specific case of the ReLU activation functions.

C. Quadratic constraint: Approximation error of the reduced-order neural network

An upper bound for the approximation error between the full and reduced-order networks can also be expressed as a quadratic constraint. This error bound will be used as a performance metric to gauge how well the reduced-order neural network approximates the full-order one, as in how well $g(x) \approx f(x)\ \forall x \in \mathcal{X}_\infty$.

Definition 3 (Approximation error): For some $\gamma_x \geq 0$, $\gamma \geq 0$, the reduced-order NN's approximation error is defined as the quadratic bound

$$\|f(x) - g(x)\|_2^2 \leq \gamma_x \|x\|_2^2 + \gamma, \quad \forall x \in \mathcal{X}_\infty. \tag{16}$$

In practice, this bound is computed by minimising over some weighted sum of γ_x and γ.

Note that by using equations (3) and (6) the approximation error $f(x) - g(x)$ can be written as

$$f(x) - g(x) = L\mu(x) \tag{17}$$

where

$$L = \begin{bmatrix} 0_{n_f \times n_x}, & W^f, & -\Psi^f, & b^l - \beta^\lambda \end{bmatrix} \tag{18}$$

Similarly,

$$\gamma_x \|x\|_2^2 + \gamma = \mu(x)^T \Gamma \mu(x) \tag{19}$$

where $\Gamma = \mathrm{blockdiag}(\gamma_x I_{n_x}, 0_{N\times N}, 0_{M\times M}, \gamma)$, so inequality (16) is equivalent to

$$\mu(x)^T (L^T L - \Gamma)\mu(x) \leq 0.$$

IV. Reduced-order neural network synthesis problem

This section contains the main result of the paper; an SDP formulation of the reduced-order NN synthesis problem (Proposition 1). To arrive at this formulation, a general statement of the synthesis problem is first defined in Theorem 1. This theorem characterises the search for the reduced-order neural network's parameters as minimising the worst-case approximation error for all inputs $x \in \mathcal{X}$.

Theorem 1: Assume the activation functions ϕ satisfy one or more of the properties from Definition 2. With fixed weights $\{w_1, w_2\} \geq 0$, if there exists a solution to

$$\min_{\Psi, \Psi_0, \Psi^l, \beta, \beta^\lambda, \mathbf{T}^i, \mathbf{T}_r^i, \tau_{x_\infty}, \gamma_x, \gamma} \quad w_1\gamma_x + w_2\gamma, \tag{20a}$$

s.t.

$$\Pi_\infty + \Lambda + L^T L - \Gamma \leq 0, \tag{20b}$$

$$\gamma_x \geq 0,\ \gamma \geq 0,$$

then the worst-case approximation error is bounded by $\|f(x) - g(x)\|_2^2 \leq \gamma_x \|x\|_2^2 + \gamma$ for all $x \in \mathcal{X}_\infty$.

Proof. See Appendix 2.

The main issue with Theorem 1 is verifying inequality (20b) since it includes a non-convex bilinear matrix inequality (BMI) between the matrix variables of the reduced-order network's weights, its biases and the scaling variables in Λ. The following proposition details how this constraint

can be written (after the application of a convex relaxation of the underlying BMI) as an LMI. The search over the reduced NN variables can then be translated into a SDP, a class of well understood convex optimisation problems with many standard solvers such as MOSEK [1] implemented through the YALMIP [28] interface in MATLAB or even the Robust Control Toolbox.

Proposition 1: Consider the full-order neural network of (2) mapping $x \to f(x)$ and the reduced-order neural network of (5) mapping $x \to g(x)$. For fixed weights $\{w_1, w_2\} \geq 0$, if there exists matrix variables $\mathbf{T}^i$ (of appropriate dimension and property[1]), $\mathbf{F}_\Psi \in \mathbb{R}^{M \times M}$, $\mathbf{F}_0 \in \mathbb{R}^{M \times n_x}$, $\mathbf{F}_\beta \in \mathbb{R}^M$, $\Psi^f \in \mathbb{R}^{n_f \times M}$ and $\beta^\lambda \in \mathbb{R}^{n_f}$ that solve

$$\min_{\mathbf{F}_\Psi, \mathbf{F}_0, \mathbf{F}_\beta, \Psi^f, \beta^\lambda, \mathbf{T}^i, \mathbf{T}^i_\tau, \tau_{x_\infty}, \gamma_x, \gamma} \quad w_1\gamma_x + w_2\gamma, \tag{21a}$$

$$\text{s.t.} \quad \Omega_{\text{Schur}} \prec 0,\ \gamma_x \geq 0,\ \gamma \geq 0, \tag{21b}$$

with Ω_{Schur} defined in (35) of Appendix 3, then the reduced-order network with weights and affine terms defined by

$$\Psi_0 = U_1^{-1}\mathbf{F}_0 \quad \Psi = U_2^{-1}\mathbf{F}_\Psi \quad \beta = U_3^{-1}\mathbf{F}_\beta, \tag{22}$$

ensures that the worst-case approximation error bound of the reduced-order neural network satisfies $\|f(x) - g(x)\|_2^2 \leq \gamma_x \|x\|_2^2 + \gamma$ for all $x \in \mathcal{X}_\infty$.

Proof. See Appendix 3.

Appendix 4 details how the matrix Λ, which characterises how the activation functions are included within the robustness condition $\Omega_{\text{Schur}} \succ 0$, can be written as a linear function of the matrix variables as required by Proposition 1 for the special case where $\phi(y) = \text{ReLU}(y)$. Some remarks about the proposition are given in Appendix 5.

Remark 2: There is some degree of flexibility in choosing the architecture of the reduced-order neural network in Proposition 1. This flexibility is viewed as an advantage of the method, as it increases its applicability, but it is also acknowledged that it could make finding the optimal architecture more challenging. In practice, it has been observed that a suitable way to fix the architecture is to set the activation function to be the same as that of the full-order network and also to set the layer dimensions of the reduced-order network to be quite low. Upon solving Proposition 1, the user should then inspect the performance of the reduced-order network generated by Proposition 1. If a satisfactory level of performance has been achieved (measured either through the bounds or from inspecting the error to the full-order network directly), then this architecture should be retained. If not, then the dimension of the reduced-order network should be increased, and Proposition 1 run again. This process should be repeated until the performance standards have been met. Algorithm 1 shows pseudo-code for this design process. ⋆

Remark 3: The reduced-order neural network generated by Proposition 1 could also be fine-tuned, as often applied to pruned neural networks. ⋆

[1]From Lemma 1 the matrices $\mathbf{T}^i$ and $\mathbf{T}^i_\tau$ may have special properties such that they must have positive elements or be diagonal.

Algorithm 1 Update the reduced-order neural network architecture

Require: Full-order neural network parameters (W, b), reduced-order neural network dimensions (m_k, λ) and tolerances ε_1, ε_2.

```
for j = 1, 2, ..., J do
    [Ψ, β] ← Proposition 1(m_k, λ)
    p_j = γ_x ||x||_2^2 + γ
    q_j = max ||f(x) − g(x)||_2^2 ∀x ∈ X'_∞ ⊆ X_∞
    if p_j ≤ ε_1 and/or q_j ≤ ε_2 then
        Break
    else
        Increase m_k and/or λ.
    end if
end for
```

V. Numerical example

The proposed reduced-order neural network synthesis method was then evaluated in two numerical examples. In both cases, the performance of the synthesized neural networks were evaluated graphically (see Figures 2-3) to give a better representation of the robustness of the approximations (the focus of this work). Only academic examples were considered due to the well-known scalability issues of SDP solvers (but which are becoming less of an issue [8]) and because performance was measured graphically. The code for the numerical examples can be obtained on request from the authors.

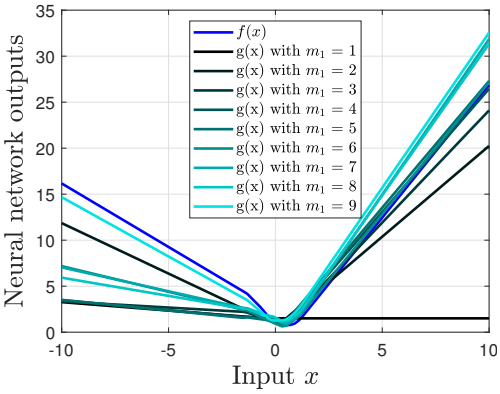
The first example explores the impact of reducing the dimension of the reduced-order neural network on its accuracy. In this case, the full-order neural network considered was a single hidden layer network of dimension 10 with the weights W^0, W^1, b^0 and b^l all obtained from sampling a zero mean normal distribution with variance 1 and which mapped a single input to a single output, $n_x = n_f = 1$, with the input constrained to $x \in [-10, 10]$. The ReLU was taken as the activation function of both the full and reduced-order neural networks. Reduced-order feed-forward neural networks with single hidden layers of various dimensions m_1 were then synthesized using Proposition 1. Figure 2a shows the various approximations obtained and Figure 2b shows how the error bounds and approximation errors changed as the dimension of the reduced-order network m_1 increased. The error bound was satisfied in all cases (albeit conservatively) and dropped as the degree of the reduced-order network increased. Even though the error bound (the blue line in the figure) from the solution of Proposition 1 monotonically decreased as the dimension of the reduced-order neural network increased, this did not imply that the actual observed worst-case error (black line in the figure) would also monotonically decrease, as observed at $m_1 = 2$. The non-monotonicity of the error highlights how the performance of several candidates reduced-order neural network architectures should be evaluated prior to deployment before an "optimal" architecture is implemented, with Algorithm 1 illustrating

one approach to conduct this search

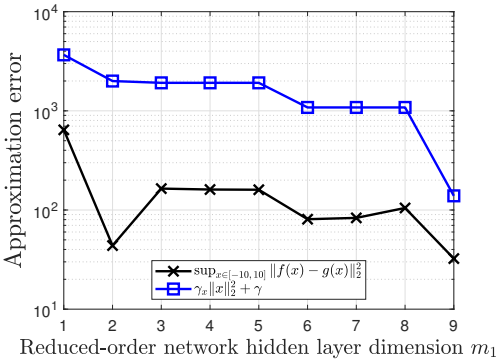
The second example considers a more complex function to approximate and illustrates some potential pitfalls of pruning too hard. In this case, the full-order network's weights were defined by $\ell = 4$, $n_k = 4$, $n_x = 1$ and with $v = (1, \ldots, n_k + 1)/n_k$ then the weights and biases were $W^0 = \cos(2\pi v)$, $b_0 = 0$, $W^k = \frac{1}{k+1}\left(\cos(2\pi v) \times \sin(2\pi v)^T\right)$, $b^k = \frac{1}{k+1}(\sin(2\pi v))$, $W^\ell = \sin(2\pi v)$ and $b^\ell = 0$. Figure 3 shows the output generated from a $\lambda = m_k = 3$ reduced-order feed-forward neural network as well as the network generated by setting the Λ_p matrices in Lemma 2 to be diagonal (this reduced the compute time but, as shown, can alter the obtained function). Also shown is the case when the full-order neural network has been pruned to have a similar number of connections as the reduced-order one by removing the 32 smallest (out of a total of 56) weights. In this case, the pruned network was cut so far that it simply generated a constant function, but further fine-tuning of the pruned network may recover performance. Likewise, fine-tuning of the reduced-order neural network (through different substitutions of J_1 and J_2, which was set to zero for these examples, or from simply applying the standard fine-tuning update of pruning) may improve the approximation of the synthesized reduced-order neural networks.

Conclusions

A method to synthesize the weights and biases of reduced-order neural networks (having few neurons) approximating the input/output mapping of a larger was introduced. A semi-definite program was defined for this synthesis problem that directly minimised the worst-case approximation error of the reduced-order network with respect to the larger one, with this error being bounded. By including the worst-case approximation error directly within the training cost function, it is hoped that the ideas explored in this paper will lead to more robust and reliable reduced-order neural network approximations. Several open problems still remain to be explored, most notably in reducing the conservatism of the bounds, scaling up the method to large neural networks and exploring the convexification of the bi-linear matrix inequality of the synthesis problem.



(a) Outputs of the full-order $f(x)$ (blue) and reduced-order $g(x)$ (grey) neural networks of various dimensions m_1.



(b) Error bounds as a function of m_1. The worst-case error between the full and reduced order neural networks is shown in black whilst the bounds obtained from Proposition 1 is shown in blue.

Fig. 2: Evaluation of the reduced-order neural networks synthesized from Proposition 1 in the first numerical example.

Acknowledgements

The authors were funded for this work through the Nextrode Project of the Faraday Institution (EPSRC Grant EP/M009521/1) and a UK Intelligence community fellowship from the Royal Academy of Engineering.

References

[1] E. D. Andersen and K. D. Andersen, "The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm," in *High Performance Optimization*. Springer, 2000, pp. 197–232.

[2] D. Angeli, "Convergence in networks with counterclockwise neural dynamics," *IEEE Transactions on Neural Networks*, vol. 20, no. 5, pp. 794–804, 2009.

[3] N. E. Barabanov and D. V. Prokhorov, "Stability analysis of discrete-time recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 292–303, 2002.

[4] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *arXiv preprint arXiv:2003.03033*, 2020.

[5] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[6] Y.-C. Chu and K. Glover, "Bounds of the induced norm and model reduction errors for systems with repeated scalar nonlinearities," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 471–483, 1999.

[7] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," *arXiv preprint arXiv:1412.7024*, 2014.

[8] S. Dathathri, K. Dvijotham, A. Kurakin, A. Raghunathan, J. Uesato, R. R. Bunel, S. Shankar, J. Steinhardt, I. Goodfellow, P. S. Liang *et al.*, "Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[9] J. Doyle, K. Glover, P. Khargonekar, and B. Francis, "State-space solutions to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems," in *Procs. of the American Control Conference*. IEEE, 1988, pp. 1691–1696.

[10] L. El Ghaoui, F. Gu, B. Travacca, and A. Askari, "Implicit deep learning," *arXiv preprint arXiv:1908.06315*, 2019.

[11] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *arXiv preprint*

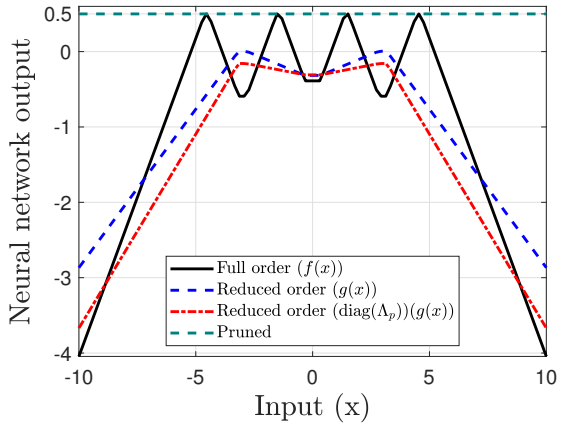


Fig. 3: Outputs of the second numerical example comparing the reduced order neural networks (with both full and diagonal Λ_p matrices in Lemma 2) and a non-fine tuned pruned neural network of an equivalent size (with the 32 smallest weights being set to zero). With this level of reduction, the pruned network gave a constant output of 1/2 whereas the reduced-order network could capture some of the variation of the function.

arXiv:1903.01287, 2019.

[12] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," *arXiv preprint arXiv:1902.09574*, 2019.

[13] C. Global, "Cloud index: Forecast and methodology, 2016–2021, white paper. [online]," *Available: https://www.cisco.com/c/en/us/ solutions/collateral/service-provider/globalcloud-index-gci/white-paper-c11-738085.htm*.

[14] K. Glover, "All optimal Hankel-norm approximations of linear multivariable systems and their L_∞-error bounds," *International Journal of Control*, vol. 39, no. 6, pp. 1115–1193, 1984.

[15] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[16] ——, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, pp. 1135–1143, 2015.

[17] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems*, 1993, pp. 164–171.

[18] S. A. Janowsky, "Pruning versus clipping in neural networks," *Physical Review A*, vol. 39, no. 12, p. 6600, 1989.

[19] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 239–242, 1990.

[20] H. K. Khalil and J. W. Grizzle, *Nonlinear Systems*, 3rd ed. Prentice hall Upper Saddle River, NJ, 2002.

[21] S. N. Kumpati, P. Kannan *et al.*, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[22] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, 1990, pp. 598–605.

[23] N. Lee, T. Ajanthan, S. Gould, and P. H. Torr, "A signal propagation perspective for pruning neural networks at initialization," *arXiv preprint arXiv:1906.06307*, 2019.

[24] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[25] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *International conference on machine learning*, 2016, pp. 2849–2858.

[26] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Advances in Neural Information Processing Systems*, 2017, pp. 2181–2191.

[27] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[28] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *International Conference on Robotics and Automation*. IEEE, 2004, pp. 284–289.

[29] A. Megretski and A. Rantzer, "System analysis via integral quadratic constraints," *IEEE Transactions on Automatic Control*, vol. 42, no. 6, pp. 819–830, 1997.

[30] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[31] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing Systems*, 1989, pp. 107–115.

[32] ——, "Using relevance to reduce network size automatically," *Connection Science*, vol. 1, no. 1, pp. 3–16, 1989.

[33] A. Raghunathan, J. Steinhardt, and P. S. Liang, "Semidefinite relaxations for certifying robustness to adversarial examples," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 877–10 887.

[34] S. Shin, K. Hwang, and W. Sung, "Fixed-point performance analysis of recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2016, pp. 976–980.

[35] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," *arXiv preprint arXiv:1511.06488*, 2015.

[36] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[37] G. Valmorbida, R. Drummond, and S. R. Duncan, "Regional analysis of slope-restricted Lurie systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 3, pp. 1201–1208, 2018.

[38] J. G. VanAntwerp and R. D. Braatz, "A tutorial on linear and bilinear matrix inequalities," *Journal of Process Control*, vol. 10, no. 4, pp. 363–385, 2000.

[39] G. Zames and P. Falb, "Stability conditions for systems with monotone and slope-restricted nonlinearities," *SIAM Journal on Control*, vol. 6, no. 1, pp. 89–108, 1968.

[40] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

Appendices

Appendix 1: Quadratic Constraint for the ReLU

Consider the generalised quadratic constraint of Lemma 2 with the $\phi(y)$ =ReLU(y) activation function and define both $\xi = W\check{x} + W_0 x + b$ and $\zeta = \Psi\check{z} + \Psi_0 x + \beta$. From Table I, it is clear that the *full order* network activation function satisfies the following quadratic constraints:

$$2\phi(\xi)^T \mathbf{T}^0(\xi - \phi(\xi)) \geq 0, \qquad \mathbf{T}^0 \in \mathbb{D}^N, \tag{23}$$
$$2(\mathbf{T}^+)^T\phi(\xi) \geq 0, \qquad \mathbf{T}^+ \in \mathbb{R}^N_+, \tag{24}$$
$$2(\mathbf{T}^{c+})^T(\phi(\xi) - \xi) \geq 0, \qquad \mathbf{T}^{c+} \in \mathbb{R}^N_+. \tag{25}$$

Similarly, the reduced order activation functions satisfy the following quadratic constraints

$$2\phi(\zeta)^T \mathbf{T}^0_r(\zeta - \phi(\zeta)) \geq 0, \qquad \mathbf{T}^0_r \in \mathbb{D}^M, \tag{26}$$
$$2(\mathbf{T}^+_r)^T\phi(\zeta) \geq 0, \qquad \mathbf{T}^+_r \in \mathbb{R}^M_+, \tag{27}$$
$$2(\mathbf{T}^{c+}_r)^T(\phi(\zeta) - \zeta) \geq 0, \qquad \mathbf{T}^{c+}_r \in \mathbb{R}^M_+. \tag{28}$$

In addition both the full and reduced order activation functions satisfy the sector constraint, but the complementarity constraint is more general so the sector constraint is redundant and thus not included. Finally, the full

and reduced order activation functions satisfy the cross constraints:

$$2(\phi(\xi)-\xi)^T\mathbf{T}^{\times}\phi(\zeta)\geq 0,\quad \mathbf{T}^{\times}\in\mathbb{R}_+^{N\times M}, \tag{29}$$

$$2(\phi(\zeta)-\zeta)^T\mathbf{T}_r^{\times}\phi(\xi)\geq 0,\quad \mathbf{T}_r^{\times}\in\mathbb{R}_+^{M\times N}. \tag{30}$$

These constraints can be combined as shown in equation (31) and then, using the definitions of $\xi,\zeta,\phi(\xi)$ and $\phi(\zeta)$ given earlier, can be expressed as given in equation (32). Note that the slope constraints of ReLU$(\cdot)$ could have also been exploited as well.

Appendix 2: Proof of Theorem 1

Inequality (20b) can be pre and post multiplied by $\mu(x)^T$ and $\mu(x)$ respectively, and then split into three components:

$$\underbrace{\begin{bmatrix}x\\1\end{bmatrix}^T P_{\mathcal{X}_\infty}\begin{bmatrix}x\\1\end{bmatrix}}_{\theta_x}+\underbrace{\mu(x)^T\Lambda\mu(x)}_{\theta_\phi}+\underbrace{\mu(x)^T(L^TL-\Gamma)\mu(x)}_{\theta_{\|f-g\|_2^2}}\leq 0. \tag{33}$$

If $x\in\mathcal{X}_\infty$, then $\theta_x\geq 0$ from Lemma 1. Also, if the nonlinear activation functions satisfy the quadratic constraint of Lemma 2, then $\theta_\phi\geq 0$. Thus, inequality (20b) implies

$$\mu(x)(L^TL-\Gamma)\mu(x)\leq 0\quad\forall\mu(x)$$

which then implies the error bound $\|f(x)-g(x)\|_2^2\leq\gamma_x\|x\|_2^2-\gamma$ holds for all $x\in\mathcal{X}_\infty$.

Appendix 3: Proof of Proposition 1

Theorem 1 requires the following matrix inequality to hold

$$\Pi_\infty+\Lambda-\Gamma+L^TL\leq 0.$$

Using the Schur complement, a sufficient condition for this to hold is

$$\Omega_{\text{mod}}=\begin{bmatrix}\Pi_\infty+\Lambda-\Gamma & L^T\\ L & -I\end{bmatrix}\leq 0.$$

This matrix is not linear due to the fact that the matrix $\Lambda(\mathbf{T}^i,\mathbf{T}_r^i,\Psi,\Psi_0,\beta)$ contains products of the constraint matrix variables $\mathbf{T}^i$ and the reduced-order network paraters Ψ,Ψ_0 and β. However, Λ can be re-written as in equation (34) where some of the matrix variables $\mathbf{T}_r^i$ are written as products of other matrix variables and two constant matrices $J_1\in\mathbb{R}^{N\times M}$ and $J_2\in\mathbb{R}^M$ which are chosen by the user. In equation (34), the $\tilde{\Lambda}_{ij}$ elements are affine functions of the $\mathbf{T}^i,\mathbf{T}_r^i$ matrix variables and U_k are matrix variables constructed from the sum of one or more $\mathbf{T}_r^i$. Defining

$$\mathbf{F}_0=U_1\Psi_0,\quad \mathbf{F}_\Psi=U_2\Psi,\quad \mathbf{F}_\beta=U_3\beta,$$

and using the expression for Λ in Ω_{schur} yields the linear matrix inequality of (35).

Once inequality (35) is satisfied, the parameters of the reduced order network can be recovered via

$$\Psi_0=U_1^{-1}\mathbf{F}_0,\quad \Psi=U_2^{-1}\mathbf{F}_\Psi,\quad \beta=U_3^{-1}\mathbf{F}_\beta.$$

Appendix 4: The matrix Λ for the case $\phi(y)=$ReLU(y)

When the activation functions of the neural network are $\phi(y)=\text{ReLU}(y)$, then the matrix $\Lambda=\Lambda_{\text{ReLU}}$ in Ω_{schur} is given as in inequality (32). Ω_{schur} becomes an LMI if Λ can be made linear. Λ_{ReLU} of (32) features the matrix variables $\mathbf{T}^0\in\mathbb{D}^N$, $\mathbf{T}_r^0\in\mathbb{D}^M$, $\mathbf{T}^+\in\mathbb{R}_+^N$, $\mathbf{T}_r^+\in\mathbb{R}_+^M$, $\mathbf{T}^{c+}\in\mathbb{R}_+^N$, $\mathbf{T}_r^{c+}\in\mathbb{R}_+^M$, $\mathbf{T}^{\times}\in\mathbb{R}_+^{N\times M}$, $\mathbf{T}_r^{\times}\in\mathbb{R}_+^{N\times M}$, as well as $\Psi\in\mathbb{R}^{M\times M}$, $\Psi_0\in\mathbb{R}^{M\times n_x}$ and $\beta\in\mathbb{R}^N$. To make Λ_{ReLU}, specific structures for $\mathbf{T}_r^{\times}$ and $\mathbf{T}_r^{c+}$ must be chosen, viz,

$$\mathbf{T}_r^{\times}=\mathbf{T}_r^0J_1,\quad \mathbf{T}_r^{c+}=\mathbf{T}_r^0J_2, \tag{36}$$

where $J_1\in\mathbb{R}^{M\times N}$ and $J_2\in\mathbb{R}^M$, which makes the substitutions

$$\mathbf{F}_\Psi^T=\Psi^T\mathbf{T}_r^0,\quad \mathbf{F}_0^T=\Psi_0^T\mathbf{T}_r^0,\quad \mathbf{F}_\beta^T=\beta^T\mathbf{T}_r^0. \tag{37}$$

The arising expression for Λ_{ReLU} is shown in equation (40) and is clearly linear in the matrix variables $T^0,\mathbf{T}_r^0,T^+,T_r^+,\mathbf{T}^{c+},\mathbf{T}^{\times},\mathbf{F}_\Psi,\mathbf{F}_0$ and $\mathbf{F}_\beta$. As in the general case, the reduced order parameters can be determined via

$$\Psi_0=(\mathbf{T}_r^0)^{-1}\mathbf{F}_0,\quad \Psi=(\mathbf{T}_r^0)^{-1}\mathbf{F}_\Psi,\quad \beta=(\mathbf{T}_r^0)^{-1}\mathbf{F}_\beta. \tag{38}$$

The scaling matrices J_p, $p\in\{1,2\}$ are constant matrices that can be picked by the user, under the stipulation that they preserve the properties of the multipliers of Lemma 2. In this work, the choice was

$$J_1=\begin{bmatrix}I_M & \mathbf{0}_{M\times(N-M)}\end{bmatrix},\ J_2=\mathbf{1}_M, \tag{39}$$

but more refined choices may also exist.

In this way, the non-convexity of the bilinear matrix inequality of the problem has been relaxed into a convex linear one. However, the substitution (37) limits the space of solutions that can be searched over by the synthesis SDP, resulting in only local optima being achieved and increased conservatism in the approximation error bounds.

Remark 4: The use of the specific structures presented in (36) is central to expressing the results in linear matrix inequality form. However, it is vital to ensure the stipulated properties of the matrices are satisfed. For instance $\mathbf{T}_r^{\times}$ is required to have all of its elements positive (or zero). This will indeed be the case if $J_1\in\mathbb{R}_+^{M\times N}$ and if $\mathbf{T}_r^0\in\mathbb{D}_+^M$ as required by Lemma 1. However, to recover the reduced order network parameters $\mathbf{T}_r^0$ also needs to be nonsingular (see equation (38) , so in the arising optimisation problem it may be prudent to choose $\mathbf{T}_r^0$ to be strictly positive definite to guarantee this. Similar comments apply to the the vector J_2. ⋆

Appendix 5: Some remarks about Proposition 1

a) Neural network synthesis: A key feature of Proposition 1 is that the parameters of the reduced-order neural network are synthesized in one shot from the solution to (21). Directly minimising the worst-case approximation error of the reduced-order neural networks may lead to more robust and reliable out-of-sample performance.

$$\begin{bmatrix} \xi \\ \zeta \\ \phi(\xi) \\ \phi(\zeta) \\ 1 \end{bmatrix}^T \begin{bmatrix} 0_{N\times N} & 0_{N\times M} & \mathbf{T}^0 & -\mathbf{T}^\times & -\mathbf{T}_{c+} \\ \star & 0_{M\times M} & -\mathbf{T}_r^\times & \mathbf{T}_r^0 & -\mathbf{T}_r^{c+} \\ \star & \star & -2\mathbf{T}^0 & \mathbf{T}^\times + (\mathbf{T}_r^\times)^T & \mathbf{T}^+ + \mathbf{T}^{c+} \\ \star & \star & \star & -2\mathbf{T}_r^0 & \mathbf{T}_r^+ + \mathbf{T}_r^{c+} \\ \star & \star & \star & \star & 0 \end{bmatrix} \begin{bmatrix} \xi \\ \zeta \\ \phi(\xi) \\ \phi(\zeta) \\ 1 \end{bmatrix} \geq 0 \tag{31}$$

$$\mu(x)^T \underbrace{\begin{bmatrix} 0 & W_0^T\mathbf{T}^0 - \Psi_0^T\mathbf{T}_r^\times & \Psi_0^T\mathbf{T}_r^0 - W_0^T\mathbf{T}^\times & -W_0^T\mathbf{T}^{c+} - \Psi_0^T\mathbf{T}_r^{c+} \\ \star & -2\mathbf{T}^0 + \mathbf{T}^0W - W^T\mathbf{T}^0 & (I_N - W)^T\mathbf{T}^\times + (\mathbf{T}_r^\times)^T(I_M - \Psi) & \mathbf{T}^0b + \mathbf{T}^+ + (I_N - W)^T\mathbf{T}^{c+} - (\mathbf{T}_r^\times)^T\beta \\ \star & \star & -2\mathbf{T}_r^0 + \mathbf{T}_r^0\Psi + \Psi^T\mathbf{T}_r^0 & \mathbf{T}_r^0\beta + \mathbf{T}_r^+ + (I_M - \Psi)^T\mathbf{T}_r^{c+} - (\mathbf{T}^\times)^Tb \\ \star & \star & \star & -(\mathbf{T}^{c+})^Tb - b^T\mathbf{T}^{c+} - (\mathbf{T}_r^{c+})^T\beta - \beta^T\mathbf{T}_r^{c+} \end{bmatrix}}_{\Lambda_{\text{ReLU}}} \mu(x) \geq 0 \tag{32}$$

$$\Lambda = \begin{bmatrix} 0 & \tilde{\Lambda}_{12} + \Psi_0^TU_1J_1^T & \tilde{\Lambda}_{13} + \Psi_0^TU_1 & \tilde{\Lambda}_{14} + \Psi_0^TU_1J_2 \\ \star & \tilde{\Lambda}_{22} & \tilde{\Lambda}_{23} + J_1U_2\Psi & \tilde{\Lambda}_{24} + J_1U_3\beta \\ \star & \star & \tilde{\Lambda}_{33} + U_2\Psi + \Psi^TU_2 & \tilde{\Lambda}_{34} + \Psi^TU_2J_2 + U_3\beta \\ \star & \star & \star & \tilde{\Lambda}_{44} + \beta^TU_3J_2 + J_2^TU_3\beta \end{bmatrix} \tag{34}$$

$$\Omega_{\text{schur}} = \begin{bmatrix} -\tau_{x_\infty} - \gamma_x I_{n_x} & \tilde{\Lambda}_{12} + \mathbf{F}_0^TJ_1^T & \tilde{\Lambda}_{13} + \mathbf{F}_0^T & \tilde{\Lambda}_{14} + \mathbf{F}_0^TJ_2 + \frac{\tau_{x_\infty}}{2}(\underline{x} + \bar{x}) & 0_{n_x\times n_f} \\ \star & \tilde{\Lambda}_{22} & \tilde{\Lambda}_{23} + J_1\mathbf{F}_\Psi & \tilde{\Lambda}_{24} + J_1\mathbf{F}_\beta & (W^f)^T \\ \star & \star & \tilde{\Lambda}_{33} + \mathbf{F}_\Psi + \mathbf{F}_\Psi^T & \tilde{\Lambda}_{34} + \mathbf{F}_\Psi^TJ_2 + \mathbf{F}_\beta & -(\Psi^f)^T \\ \star & \star & \star & \tilde{\Lambda}_{44} + \mathbf{F}_\beta^TJ_2 + J_2^T\mathbf{F}_\beta - \gamma - \underline{x}^T\tau_{x_\infty}\bar{x} & b^l - \beta^\lambda \\ \star & \star & \star & \star & -I_{n_f} \end{bmatrix} \tag{35}$$

$$\Lambda_{\text{ReLU}} = \begin{bmatrix} 0 & W_0^T\mathbf{T}^0 - \mathbf{F}_0^TJ_1^T & \mathbf{F}_0^T - W_0^T\mathbf{T}^\times & -W_0^T\mathbf{T}^{c+} - \mathbf{F}_0^TJ_2 \\ \star & -2\mathbf{T}^0 + \mathbf{T}^0W - W^T(\mathbf{T}^0)^T & (I_N - W)^T\mathbf{T}^\times + J_1\mathbf{T}_r^0 - J_1\mathbf{F}_\Psi & \mathbf{T}^0b + \mathbf{T}^+ + (I - W)^T\mathbf{T}^{c+} - J_1\mathbf{F}_\beta \\ \star & \star & -2\mathbf{T}_r^0 + \mathbf{F}_\Psi + \mathbf{F}_\Psi^T & \mathbf{F}_\beta + \mathbf{T}_r^+ + \mathbf{T}_r^0J_2 - \mathbf{F}'_\Psi J_2 - (\mathbf{T}^\times)^Tb \\ \star & \star & \star & -(\mathbf{T}^{c+})^Tb - b^T\mathbf{T}^{c+} - J_2^T\mathbf{F}_\beta - \mathbf{F}_\beta^TJ_2 \end{bmatrix} \tag{40}$$

b) Computational cost: The main source of computational complexity in Proposition 1 is the growth in the number of decision variables as the problem involves matrix variables. This limits the applicability of the proposed approach to modestly size networks. However, this issue could be reduced by imposing sparsity on the various matrix variables, such as restricting the scaling Λ_p matrices in Lemma 2 to be diagonal or sparse. Scalability issues are a common curse of methods providing robustness guarantees, like [11], but methods are being developed to alleviate these issues, e.g. [8].

c) Bilinearity: The BMI constraint in (40) is the source of non-convexity in the problem [38] which had to be relaxed. It is highly likely that, for a given full-order neural network, there would exist better substitutions than (39), however, (39) seemed to work quite well in the numerical example of Section V.

d) Robust approximation: Since the robustness analysis holds for all nonlinear activation functions satisfying the quadratic inequalities of Lemma 1 and all inputs $x \in \mathcal{X}_\infty$, the performance guarantees of Proposition 1 may be conservative.

Ross Drummond completed a DPhil degree in the Control Group at the University of Oxford in 2017 where he then held a post-doctoral research assistant position until 2020. He is currently a UK Intelligence Community Research Fellow sponsored by the Royal Academy of Engineering modelling thermal runaway in lithium ion battery packs. His main research interests include modelling electrochecmical energy storage devices, developing stability criteria for non-linear systems, characterising externally positive linear systems and providing neural network robustness certificates.

Matthew Turner was born in Corby, England in 1975. He received the BEng degree in Electrical and Electronic Engineering in 1996, from the University of Surrey, and the PhD in Control Engineering in 2000, from the University of Leicester. He remained at Leicester as a post-doc and then took up a lectureship, before being promoted to senior lecturer and then professor. In 2020, he moved to the University of Southampton as a professor in the cyber physical systems research group. His main research interests are in robust and nonlinear control theory and their application to a range of physical systems. Recently, he has become interested in adaptive and learning-based control systems, particularly in their potential advantages and also their fragility.

Stephen Duncan is a Professor in the Department of Engineering Science at the University of Oxford, where he is a member of the Control group. Prior to joining Oxford, he was Reader in the Control Systems Centre at the University of Manchester. He has an MA in Physics from the University of Cambridge and an MSc and PhD in Control Systems from Imperial College, London. His research concentrates on the design and implementation of feedback control systems for a range of applications, including energy storage, synchrotrons, manufacturing processes and transport.