Provably Correct Physics-Informed Neural Networks

Francisco Eiras [1] **Adel Bibi** [1] **Rudy Bunel** [2] **Krishnamurthy Dj Dvijotham** [2] **Philip H.S. Torr** [1]
M. Pawan Kumar [2]

Abstract

Physics-informed neural networks (PINN) have been proven efficient at solving partial differential equations (PDE). However, previous works have failed to provide guarantees on the *worst-case* residual error of a PINN across the spatio-temporal domain – a measure akin to the tolerance of numerical solvers – focusing instead on point-wise comparisons between their solution and the ones obtained by a solver at a set of inputs. In real-world applications, one cannot consider tests on a finite set of points to be sufficient grounds for deployment. To alleviate this issue, we establish tolerance-based *correctness* conditions for PINNs over the entire input domain. To verify the extent to which they hold, we introduce ∂-CROWN: a general and efficient post-training framework to bound PINN errors. We demonstrate its effectiveness in obtaining tight certificates by applying it to two classical PINNs – Burgers' and Schrödinger's equations –, and two more challenging ones – the Allan-Cahn and Diffusion-Sorption equations.

1. Introduction

While partial differential equations (PDE) are key to simulating complex physical systems, obtaining accurate solutions at an appropriate spatio-temporal resultion is challenging (Kochkov et al.). Inspired by the success of machine learning in other domains, recent work has attempted to overcome this challenge through *physics-informed neural networks* (PINN) (Raissi et al., 2019a; Sun et al., 2020; Pang et al., 2019). For example, the Diffusion-Sorption equation – with real-world applications in the modeling of groundwater contaminant transport – takes 59.83s to solve per inference point using a classical PDE solver, while the PINN version from Takamoto et al. (2022) takes only 2.7×10^{-3}s, a speed-up of more than 10^4 times.

After training, PINN accuracy is usually empirically estimated by measuring the solution outputs over a discrete set of inputs and comparing them to standard numerical PDE solvers. Crucially, most current work on PINNs provides no formal correctness guarantees that are applicable for *every* input within the feasible domain. While testing on a finite set of points provides a good initial signal for accuracy, this cannot be relied upon in practice as the performance could be substantially worse on a different set of inputs. In order to alleviate the deficiencies of previous evaluation criteria, we introduce formal, tolerance-based *correctness* conditions for PINNs, as well as a post-training framework to certify the degree to which these hold.

Our contributions are threefold. **(i)** We formally define global correctness conditions for general PINNs that approximate solutions of PDEs. **(ii)** We introduce a general, efficient, and scalable *correctness certification framework* (∂-CROWN) to theoretically verify PINNs over their entire spatio-temporal domains. **(iii)** We demonstrate our framework on two widely studied PDEs in the context of PINNs, Burgers' and Schrödinger's equations (Raissi et al., 2019a), and two more challenging ones with real-world applications, the Allan-Cahn equation (Monaco & Apiletti, 2023) and the Diffusion-Sorption equation (Takamoto et al., 2022).

2. Preliminaries and Related work

Given vector $\mathbf{a} \in \mathbb{R}^d$, $\mathbf{a}_i$ refers to its i-th component, and $\partial_{\mathbf{x}_i^j} f$ is the j-th partial derivative of a function $f : \mathbb{R}^n \to \mathbb{R}$ with respect to the i-component of its input, $\mathbf{x}_i$. We use $f(\mathbf{x})$ and f interchangeably. We take $\mathbb{L}^{(i)}_{\mathbf{W},\mathbf{b}}(\mathbf{x}) = \mathbf{W}^{(i)}\mathbf{x} + \mathbf{b}^{(i)}$ to be a function of $\mathbf{x}$ parameterized by the weights $\mathbf{W}^{(i)}$ and the bias $\mathbf{b}^{(i)}$. We define an L-layer *fully connected neural network* $g : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$ for an input $\mathbf{x} \in \mathbb{R}^{d_0}$ as $g(\mathbf{x}) = y^{(L)}(\mathbf{x})$ where $y^{(k)}(\mathbf{x}) = \mathbb{L}^{(k)}_{\mathbf{W},\mathbf{b}}(z^{(k-1)}(\mathbf{x}))$, $z^{(k-1)}(\mathbf{x}) = \sigma(y^{(k-1)}(\mathbf{x}))$, $z^{(0)}(\mathbf{x}) = \mathbf{x}$, in which $\mathbf{W}^{(k)} \in \mathbb{R}^{d_k \times d_{k-1}}$ and $\mathbf{b}^{(k)} \in \mathbb{R}^{d_k}$ are the weight and bias of layer k, σ is the nonlinear activation function, and $k \in \{1, \dots, L\}$.

Since our certification framework for PINNs is based on the verification literature of image classifiers, we divide this

*Equal contribution [1]Department of Engineering Science, University of Oxford [2]Google DeepMind. Correspondence to: Francisco Eiras <eiras@robots.ox.ac.uk>.

Presented at the 2^{nd} Workshop on Formal Verification of Machine Learning, co-located with the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA., 2023.

section in two: related work for PINNs, and previous work on the robustness verification/bounding of neural networks.

2.1. Physics-informed neural networks (PINNs)

We consider general nonlinear PDEs of the form:

$$f(t,\hat{\mathbf{x}}) = \partial_t u(t,\hat{\mathbf{x}}) + \mathcal{N}[u](t,\hat{\mathbf{x}}) = 0,\ \hat{\mathbf{x}} \in \mathcal{D}, t \in [0,T], \tag{1}$$

where f is the residual of the PDE, t and $\hat{\mathbf{x}}$ are the temporal and spatial components of the input respectively (for conciseness, $\mathbf{x} = (t,\hat{\mathbf{x}}) \in \mathcal{C}$, with $\mathbf{x}_0 = t$), $u : [0,T] \times \mathcal{D} \to \mathbb{R}$ is the solution, $\mathcal{N}$ is a nonlinear differential operator on u, $T \in \mathbb{R}^+$, and $\mathcal{D} \subset \mathbb{R}^D$. Further, the PDE is defined under (1) initial conditions, *i.e.*, $u(0,\hat{\mathbf{x}}) = u_0(\hat{\mathbf{x}})$, for $\hat{\mathbf{x}} \in \mathcal{D}$, and (2) general Robin boundary conditions, *i.e.*, $au(t,\hat{\mathbf{x}}) + b\partial_{\mathbf{n}} u(t,\hat{\mathbf{x}}) = u_b(t,\hat{\mathbf{x}})$, for $a, b \in \mathbb{R}$, $\hat{\mathbf{x}} \in \delta\mathcal{D}$ and $t \in [0,T]$, and $\partial_{\mathbf{n}} u$ is the normal derivative at the border. We assume f is the residual of an R^{th} order PDE, which can be written as $f = \mathcal{P}(u, \partial_{\mathbf{x}_0} u, \ldots, \partial_{\mathbf{x}_D} u, \ldots, \partial_{\mathbf{x}_D^R} u)$, where $\mathcal{P}$ is a nonlinear function of partial derivatives of u.

Since Raissi et al. (2019a) introduced PINNs, a variety of different ones have since emerged in a wide range of applications (Raissi et al., 2019b; Liu & Wang, 2019; Sun et al., 2020; Jin et al., 2021; Fang & Zhan, 2019; Pang et al., 2019). We focus on continuous-time PINNs (Raissi et al., 2019a) which result from approximating the solution, $u(\mathbf{x})$, using a neural network parameterized by θ, $u_\theta(\mathbf{x})$. We refer to this network as the *approximate solution*. In that context, the *physics-informed neural network* (or residual) is $f_\theta(\mathbf{x}) = \partial_t u_\theta(\mathbf{x}) + \mathcal{N}[u_\theta](\mathbf{x})$. Note f_θ has the same order as f, and can be described similarly as a nonlinear function with the partial derivatives applied to u_θ instead of u. Some previous works have focused on providing guarantees on either the training process (Shin et al., 2020; Wang et al., 2022b) or generalization errors Mishra & Molinaro (2022); Ryck & Mishra (2022), though typically under several assumptions. Our verification framework is applicable post-training to any PINN with the solution modeled by a fully connected network.

2.2. Robustness Verification and Bounding neural network outputs

The presence of adversarial examples, *i.e.*, small local input perturbations that lead to large output changes, was established by Szegedy et al. (2013) in the context of image classification. As robust classifiers emerged (Madry et al., 2017), so did attempts to certify them formally (Katz et al., 2017; Ehlers, 2017; Huang et al., 2017; Gowal et al., 2018; Mirman et al., 2018). An efficient set of methods poses the problem as a convex relaxation of the original nonlinear network and obtains closed-form solution to the bounding problem (Zhang et al., 2018; Wang et al., 2021; Xu et al., 2020b). Xu et al. (2020a) extended the linear relaxation framework from Zhang et al. (2018) to general computation graphs, but the purely backward propagation nature makes it potentially less efficient than custom bounds/hybrid approaches (Shi et al., 2020). For the sake of computational efficiency, we consider the bounds obtained using CROWN (Zhang et al., 2018)/α-CROWN (Xu et al., 2020b) as the base for our framework.

Given a fully connected network g, the goal is to compute $\max / \min_{x \in \mathcal{C}} g(\mathbf{x})$, where $\mathcal{C}$ is the applicability domain. Typically for image classifier verification, $\mathcal{C} = \mathbb{B}^p_{\mathbf{x},\epsilon} = \{\mathbf{x}' : \|\mathbf{x}' - \mathbf{x}\|_p \leq \epsilon\}$, *i.e.*, it is a *local* ℓ_p-ball of radius ϵ around an input from the test set $\mathbf{x}$. CROWN solves the optimization problem by *back-propagating* linear bounds of $g(\mathbf{x})$ through each hidden layer of the network until the input is reached, eventually obtaining:

$$\min_{\mathbf{x}\in\mathcal{C}} g(\mathbf{x}) \geq \min_{\mathbf{x}\in\mathcal{C}} \mathbf{A}^L\mathbf{x} + \mathbf{a}^L,\ \max_{\mathbf{x}\in\mathcal{C}} g(\mathbf{x}) \leq \max_{\mathbf{x}\in\mathcal{C}} \mathbf{A}^U\mathbf{x} + \mathbf{a}^U,$$

where $\mathbf{A}^L$, $\mathbf{a}^L$, $\mathbf{A}^U$ and $\mathbf{a}^U$ are computed in polynomial time from $\mathbf{W}^{(k)}, \mathbf{b}^{(k)}$, and the linear activation relaxation parameters. The solution to the optimization problems above given simple constraints $\mathcal{C}$ can be obtained in closed-form. α-CROWN (Xu et al., 2020b) improves these bounds by optimizing the linear relaxations for tightness.

3. @-CROWN: PINN Correctness Certification Framework

By definition, u_θ is a correct solution to the PINN f_θ – and therefore the PDE $f(\mathbf{x}) = 0$ – if 3 conditions are met: ① the norm of the solution error with respect to the initial condition is upper bounded within an acceptable tolerance, ② the norm of the error with respect to the boundary conditions is bounded within an acceptable tolerance, and ③ the norm of the residual is bounded within an acceptable convergence tolerance. We define these as PINN *correctness conditions*, and formalize them in Definition 1.

Definition 1 (Correctness Conditions for PINNs)**.** *$u_\theta : [0,T] \times \mathcal{D} \to \mathbb{R}$ is a $\delta_0, \delta_b, \varepsilon$-globally correct approximation of the exact solution $u : [0,T] \times \mathcal{D} \to \mathbb{R}$ if:*

① $\max_{\hat{\mathbf{x}} \in \mathcal{D}} |u_\theta(0,\hat{\mathbf{x}}) \quad u_0(\hat{\mathbf{x}})|^2 \quad \delta_0,$

② $\max_{t\in[0,T], \hat{\mathbf{x}} \in \delta\mathcal{D}} |au_\theta(t,\hat{\mathbf{x}}) + b\partial_{\mathbf{n}} u_\theta(t,\hat{\mathbf{x}}) \quad u_b(t,\hat{\mathbf{x}})|^2 \quad \delta_b,$

③ $\max_{\mathbf{x}\in\mathcal{C}} |f_\theta(\mathbf{x})|^2 \quad \varepsilon.$

In practice, δ_0, δ_b, and ε correspond to tolerances similar to the ones given by numerical solvers for f. Verifying these conditions requires bounding a linear function of u_θ for ①, bounding for a linear function of u_θ and $\partial_{\mathbf{n}} u_\theta$ for ②, and the PINN residual f_θ for ③. To achieve ① we can directly use CROWN/α-CROWN (Zhang et al., 2018; Xu et al., 2020b). However, bounding ② and ③ with a linear

Algorithm 1 Greedy Input Branching

Input: function h, input domain C, # splits N_b, # empirical samples N_s, # branches per split N_d
Result: lower bound h_{lb}, upper bound h_{ub}

$B; \mathcal{B} = ;$
$\hat{h}_{lb}; \hat{h}_{ub} = \min \max h(\text{SAMPLE}(C, N_s))$
$h_{lb}; h_{ub} = @\text{CROWN}(h; C)$
$\mathcal{B}[C] = (h_{lb}; h_{ub})$
$B \quad [C] = \max(\hat{h}_{lb} \quad h_{lb}; h_{ub} \quad \hat{h}_{ub})$
for $i \in \{1; \dots; N_b\}$ **do**
  $C_i = B.\text{POP}(\arg\max_{C'} B[C'])$
  foreach $C' \in \text{DOMAINSPLIT}(C_i, N_d)$ **do**
    $h^0_{lb}; h^0_{ub} = @\text{CROWN}(h; C')$
    $\mathcal{B}[C'] = (h^0_{lb}; h^0_{ub})$
    $B \quad [C'] = \max(\hat{h}_{lb} \quad h^0_{lb}; h^0_{ub} \quad \hat{h}_{ub})$
$h_{lb}; h_{ub} = \min_{C'} \mathcal{B}_0[C']; \max_{C'} \mathcal{B}_1[C']$
return h_{lb}, h_{ub}

Figure 1: Bounding Derivatives with @CROWN our hybrid scheme for bounding $@_{x_i} u$ and $@_{x_i^2} u$ includes back-propagation and forward substitution (inspired by Shi et al. (2020)) to compute bounds in $O(L)$ instead of the $O(L^2)$ complexity of full back-propagation as in Xu et al. (2020a).

function in x efficiently requires a method to bound linear and nonlinear functions of the partial derivatives u of

We propose @CROWN, an efficient framework to: (i) compute closed-form bounds on the partial derivatives of an arbitrary fully-connected network u, and (ii) bound a polynomial function of those partial derivative terms i.e., f. Throughout this section, we assume $u(x) = g(x)$ as per Section 2, with $d_0 = 1 + D$. Proofs for lemmas and theorems presented are in Appendix D.

(i) Bounding Partial Derivatives of u. To bound the partial derivatives of u for a PINN of order $R \quad 2$, we start by explicitly obtaining the expressions for $@_{x_i} u$ and $@_{x_i^2} u$, which we derive analytically in Appendix D and present as a computation graph in Figure 1. While we only compute the expression for the second derivative with respect to the same input variable, it would be trivial to extend it to cross derivatives (i.e., $@_{x_i x_j} u$ for $i \neq j$).

The computation of the output and intermediate pre-activation bounds for u can be done using CROWN/CROWN (Zhang et al., 2018; Xu et al., 2020b). As such, for what follows, we assume that for $x \in C$, both the bounds on u and $y^{(k)}; \forall k$ are given. Combining this with the explicit expressions of $@_{x_i} u$ and $@_{x_i^2} u$ allows us to obtain the following linear bounds.

Theorem 1 (@CROWN: Linear Bounding $@_{x_i} u$). There exist two linear functions $@_{x_i} u^U$ and $@_{x_i} u^L$ s.t. it holds $\forall x \in C$: $@_{x_i} u^L \quad @_{x_i} u \quad @_{x_i} u^U$, with the linear coefficients computed recursively in closed-form in $O(L)$ time.

Theorem 2 (@CROWN: Linear Bounding $@_{x_i^2} u$). Assume that through a previous bounding of $@_{x_i} u$, we have linear lower and upper bounds on $@_{x_i} z^{(k \quad 1)}$ and $@_{(k \quad 1)} z^{(k)}$. There exist two linear functions $@_{x_i^2} u^U$ and $@_{x_i^2} u^L$ s.t. it holds $\forall x \in C$: $@_{x_i^2} u^L \quad @_{x_i^2} u \quad @_{x_i^2} u^U$, with the coefficients computed recursively in closed-form in $O(L)$ time.

The formal statement of Theorem 1 and 2 and expressions for $@_{x_i} u^L$, $@_{x_i} u^U$, $@_{x_i^2} u^L$ and $@_{x_i^2} u^U$ are provided in Appendix D.3 and D.4. Note that these bound are not computed using fully backward propagation as in Xu et al. (2020a). Instead we use a hybrid scheme in the spirit of Shi et al. (2020) for the sake of efficiency. We perform backward propagation to compute $@_{(k \quad 1)} z^{(k)}$ as a function of $y^{(k)}$, and forward-substitute the pre-computed CROWN bounds $L^{(k);L}_{A;a}(x) \quad y^{(k)} \quad L^{(k);U}_{A;a}(x)$ at that point instead of fully backward propagating which would have $O(L^2)$ complexity. This induces a significant speed-up while achieving tight enough bounds. Figure 1 showcases the back-propagation and forward substitution paths for bounding $@_{x_i} u$ in blue, and the ones for $@_{x_i^2} u$ in green. The computation of these bounds requires relaxing $^0(y^{(k)})$ and $^{00}(y^{(k)})$.

Following a similar argument to CROWN (Zhang et al., 2018), assuming $C = \{x \in \mathbb{R}^{d_0} : x^L \quad x \quad x^U\}$, we can obtain closed-form expressions for constant global bounds on the linear functions $@_{x_i} u^U$, $@_{x_i} u^L$, $@_{x_i^2} u^U$, $@_{x_i^2} u^L$, which we formulate and prove in Appendix D.5[1].

(ii) Bounding f. With the partial derivative terms bounded, to bound f, we use McCormick envelopes (McCormick, 1976) to obtain linear lower and upper bound functions $f^L \quad f \quad f^U$:

$$f^U = \quad ^U_0 + \quad ^U_1 u + \sum_{j=1}^{R} \sum_{@_{x_i^j} \in \mathcal{N}(j)} \quad ^U_{j;i} @_{x_i^j} u;$$

and similarly for f^L, where $^U_0$, $^U_1$, and $^U_{i;j}$ are functions of the global lower and upper bounds of u and $@_{x_i^j} u$. To

[1] Note that this is different from the CROWN case in which C is assumed to be an ∞-ball around an input x.

(a) (b) (c) (d)

Figure 2: Certifying with @CROWN: time evolution of u, and the residual errors as a function of the spatial temporal domain (log-scale) $|f|$, for (a) Burgers' equation (Raissi et al., 2019b), (b) Schrödinger's equation (Raissi et al., 2019b), (c) Allan-Cahn's equation (Monaco & Apiletti, 2023), and (d) the Diffusion-Sorption equation (Takamoto et al., 2022).

obtain f^U and f^L as linear functions of x, we replace u and $\partial_{x_j} u$ with the lower and upper bound linear expressions from (i) depending on the sign of the coefficients ^{U} and ^{L}. As with the partial derivatives, since $\mathcal{C} = \{x \in \mathbb{R}^{d_0} : x^L \leq x \leq x^U\}$ we can then solve $\max_{x \in \mathcal{C}} f^U$ and $\min_{x \in \mathcal{C}} f^L$ in closed-form (Appendix D.5).

3.1. Tighter Bounds via Greedy Input Branching

Using @CROWN we can compute a bound on a nonlinear function of derivatives of u, which we will generally refer to as h, for $x \in \mathcal{C}$. However, given the approximations used in the bounding process, it is likely that such bounds will be too loose to be useful. To improve upon them, we introduce greedy input branching in Algorithm 1. The idea is to recursively divide the input domain ($\mathcal{D}$OMAIN SPLIT, line 8) - exploring the areas where the current bounds are farther from the empirical optima obtained via sampling (SAMPLE, line 2) - and globally bound the output of h as the worst-case of all the branches (line 12). As the number of splits, N_b, increases, so does the tightness of the bounds.

4. Experiments and Discussion

The aim of this experimental section is to showcase that the Definition 1 certificates obtained with @CROWN are tight compared to empirical errors computed with a large number of samples. Additional experiments are found in Appendix A, where we highlight the relationship of our residual-based certificates and the commonly reported solution errors, and in Appendix B, where we qualitatively analyze the importance of greedy input branching in the success of our method.

We apply @CROWN to two widely studied PINNs from Raissi et al. (2019a), Burgers' and Schrödinger's equations, as well as to the more complex Allen-Cahn's equation from Monaco & Apiletti (2023), and the Diffusion-Sorption equation from Takamoto et al. (2022). As required by @CROWN, we propose an algorithm in Appendix E to relax $^{\prime}$ and $^{\prime\prime}$ given pre-activation bounds. All timing results were obtained on a 10-core M1 Max MacBook Pro.

Burgers' Equation This one-dimensional PDE is used in several areas of mathematics, fluid dynamics, nonlinear acoustics, gas dynamics and traffic flow, and is derived from the Navier-Stokes equations for the velocity field by dropping the pressure gradient (Raissi et al., 2019a). It is defined for $t \in [0, 1]$ and $x \in [-1, 1]$ as:

$$\partial_t u(t,x) + u(t,x)\partial_x u(t,x) - (0.01/\pi)\partial_{x^2} u(t,x) = 0, \quad (2)$$

for $u(0,x) = -\sin(\pi x)$, $u(t,-1) = u(t,1) = 0$. The solution $u : \mathbb{R}^2 \to \mathbb{R}$ is modeled by an 8-hidden layer, 20 neurons per layer network (Raissi et al., 2019a). The training process took 13.35 minutes, and resulted in a mean ℓ_2 error of $6.1 \cdot 10^{-4}$, with a visualization in Figure 2a.

Schrödinger's Equation Schrödinger's equation is a classical field equation used to study quantum mechanical systems. In Raissi et al. (2019a), Schrödinger's equation is defined for $t \in [0, \pi/2]$ and $x \in [-5, 5]$ as:

$$i\partial_t u(t,x) + 0.5\partial_{xx} u(t,x) + |u(t,x)|^2 u(t,x) = 0, \quad (3)$$

where $u : [0, \pi/2] \times \mathbb{R} \to \mathbb{C}$ is a complex-valued solution, for initial conditions $u(0,x) = 2\,\mathrm{sech}(x)$, and periodic boundary conditions $u(t,-5) = u(t,5)$ and $\partial_x u(t,-5) = \partial_x u(t,5)$. As in Raissi et al. (2019b), $u : \mathbb{R}^2 \to \mathbb{R}^2$ is a 5-hidden layer, 100 neurons per layer network. The training took 23.67 minutes, and resulted in a mean ℓ_2 error of $1.74 \cdot 10^{-3}$, with a visualization in Figure 2b.

Table 1: Certifying with @CROWN: Monte Carlo (MC) sampled maximum values of 10^4 and 10^6 samples) and upper bounds computed using @CROWN with N_b branchings for ① initial conditions, ② boundary conditions, and ③ residual condition for (a) Burgers (Raissi et al., 2019b), (b) Schrödinger (Raissi et al., 2019b), (c) Allen-Cahn (Monaco & Apiletti, 2023), and (d) Diffusion-Sorption (Takamoto et al., 2022) equations.

	MC max (10^4)	MC max (10^6)	@CROWN u_b (time [s])
(a) Burgers (Raissi et al., 2019b)			
① $\lvert u(0,x) - u_0(x)\rvert^2$	1.59 10 6	1.59 10 6	2.63 10 6 (116.5)
② $\lvert u(t,-1) + 1\rvert^2$	8.08 10 8	8.08 10 8	6.63 10 7 (86.7)
$\lvert u(t,1) - 1\rvert^2$	6.54 10 8	6.54 10 8	9.39 10 7 (89.8)
③ $\lvert f(x,t)\rvert^2$	1.23 10 2	1.80 10 2	1.03 10 1 (2.8 10^5)
(b) Schrödinger (Raissi et al., 2019b)			
① $\lvert u(0,x) - u_0(x)\rvert^2$	7.06 10 5	7.06 10 5	8.35 10 5 (305.2)
② $\lvert u(t,5) - u(t,-5)\rvert^2$	7.38 10 7	7.38 10 7	5.73 10 6 (545.4)
$\lvert \partial_x u(t,5) - \partial_x u(t,-5)\rvert^2$	1.14 10 5	1.14 10 5	5.31 10 5 (2.4 10^3)
③ $\lvert f(x,t)\rvert^2$	7.28 10 4	7.67 10 4	5.55 10 3 (1.2 10^6)
(c) Allen-Cahn (Monaco & Apiletti, 2023)			
① $\lvert u(0,x) - u_0(x)\rvert^2$	1.60 10 3	1.60 10 3	1.61 10 3 (52.7)
② $\lvert u(t,-1) - u(t,1)\rvert^2$	5.66 10 6	5.66 10 6	5.66 10 6 (95.4)
③ $\lvert f(x,t)\rvert^2$	10.74	10.76	10.84 (6.7 10^5)
(d) Diffusion-Sorption (Takamoto et al., 2022)			
① $\lvert u(0,x)\rvert^2$	0.0	0.0	0.0 (0.2)
② $\lvert u(t,0) - 1\rvert^2$	4.22 10 4	4.39 10 4	1.09 10 3 (72.5)
$\lvert u(t,1) - D\partial_x u(t,1)\rvert^2$	2.30 10 5	2.34 10 5	2.37 10 5 (226.4)
③ $\lvert f(x,t)\rvert^2$	1.10 10 3	21.09	21.34 (2.4 10^6)

Allan-Cahn Equation The Allan-Cahn equation is a form of reaction-diffusion equation, describing the phase separation in multi-component alloy systems (Monaco & Apiletti, 2023). In 1D, it is defined for $t \in [0,1]$ and $x \in [-1,1]$ as:

$$\partial_t u(t,x) + \alpha\,(u^2(t,x) - 1)\,\partial_{x^2} u(t,x) = 0\,, \quad (4)$$

for $\alpha = 5$, $\beta = 10^{-4}$, and $u(0,x) = x^2\cos(\pi x)$, $u(t,-1) = u(t,1)$. The solution $u : R^2 \to R$ is modeled by an 6-hidden layer, 40 neurons per layer network, and due to its complexity, it is trained using the Causal training scheme from Monaco & Apiletti (2023). The training process took 18.56 minutes, and resulted in a mean ℓ_2 error of $7.9 \cdot 10^{-3}$, with a visualization in Figure 2c.

Diffusion-Sorption The diffusion-sorption equation models a diffusion system which is retarded by a sorption process, with one of the most prominent applications being groundwater contaminant transport (Takamoto et al., 2022). In (Takamoto et al., 2022), the equation is defined for $t \in [0,500]$ and $x \in [0,1]$ as:

$$\partial_t u(t,x) - D/R(u(t,x))\,\partial_{x^2} u(t,x) = 0\,; \quad (5)$$

where $D = 5 \cdot 10^{-4}$ is the effective diffusion coefficient, and $R(u(t,x))$ is the retardation factor representing the sorption that hinders the diffusion process (Takamoto et al., 2022). In particular, we consider $R(u(t,x)) = 1 + \frac{(1-\phi)}{\phi}\rho_s k n_f u^{n_f - 1}(t,x)$, where $\phi = 0.29$ is the porosity of the porus medium, $\rho_s = 2880$ is the bulk density, $k = 3.5 \cdot 10^{-4}$ is the Freundlich's parameter, and $n_f = 0.874$ is the Freundlich's exponent. The initial and boundary conditions are defined as $u(0,x) = 0$, $u(t,0) = 0$ and $u(t,1) = D\partial_x u(t,1)$. The solution $u : R^2 \to R$ is modeled by a 7-hidden layer, 40 neurons per layer network, and we obtain the trained parameters from Takamoto et al. (2022). The mean ℓ_2 solution error is $9.9 \cdot 10^{-2}$, with a visualization in Figure 2d.

@CROWN certification We verify the global correctness conditions of the PINNs by applying the framework from Section 3. We report in Table 1 our verification of the initial conditions ① using $N_b = 5\,k$ splits, boundary conditions ② using $N_b = 5\,k$ splits, and the certified bounds on the residual condition ③ using $N_b = 2\,M$ splits. We observe that @CROWN approaches the empirical bounds obtained using Monte Carlo sampling while providing the guarantee that no point within the domain breaks those bounds, effectively establishing the tolerances from Definition 1.

5. Discussion and Conclusion

We show that@CROWN is able to obtain tight upper bounds on the correctness conditions established in Definition 1. Of particular relevance is the case of the residual condition③ for the Diffusion-Sorption equation, for which varying the number of MC samples leads to distinct results - using 10^4 estimates puts the maximum at $1.10 \cdot 10^{-3}$, while 10^6 samples give an estimate of 21.09 - highlighting the need for our framework to obtain guarantees across the full domain. Note that the absolute values of the residual errors can be seen as a function of the PDE itself, and thus cannot be compared across different PINNs. As shown in Section A, they are instead connected to PDE solution errors, and can be compared within the same system. In Appendix C we study the effect of the training method from Shekarpaz et al. (2022) on empirical/certified errors.

One of the limitations of our method is the running time, which for residual verification is in the order of 10^5–10^6 for each of the PINNs studied. This is mainly due to the need to perform a high number of branchings ($2^{N_{in}}$) as a result of the looseness of the bounds obtained by@CROWN on each individual one. These issues become more accentuated as the input dimension grows, since the number of branches is expected to grow exponentially. In future work we aim to improve the tightness of the bounds to be able to apply our framework to larger, higher dimensional PINNs.

References

Ehlers, R. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pp. 269–286. Springer, 2017.

Fang, Z. and Zhan, J. Deep physical informed neural networks for metamaterial design. *IEEE Access*, 8:24506–24513, 2019.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. Safety verification of deep neural networks. In *International conference on computer aided verification*, pp. 3–29. Springer, 2017.

Jin, X., Cai, S., Li, H., and Karniadakis, G. E. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pp. 97–117. Springer, 2017.

Kim, J., Lee, K., Lee, D., Jhin, S. Y., and Park, N. Dpm: a novel training method for physics-informed neural networks in extrapolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8146–8154, 2021.

Kochkov, D., Sanchez-Gonzalez, A., Smith, J., Pfaff, T., Battaglia, P., and Brenner, M. P. Learning latent field dynamics of pdes.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

Liu, D. and Wang, Y. Multi-fidelity physics-constrained neural network and its application in materials modeling. *Journal of Mechanical Design*, 141(12), 2019.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

McCormick, G. P. Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. *Mathematical programming*, 10(1): 147–175, 1976.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3578–3586. PMLR, 2018.

Mishra, S. and Molinaro, R. Estimates on the generalization error of physics-informed neural networks for approximating pdes. *IMA Journal of Numerical Analysis*, 2022.

Monaco, S. and Apiletti, D. Training physics-informed neural networks: One learning to rule them all? *Results in Engineering*, 18:101023, 2023.

Pang, G., Lu, L., and Karniadakis, G. E. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019a.

Raissi, M., Wang, Z., Triantafyllou, M. S., and Karniadakis, G. E. Deep learning of vortex-induced vibrations. Journal of Fluid Mechanics, 861:119–137, 2019b.

Ryck, T. D. and Mishra, S. Generic bounds on the approximation error for physics-informed (and) operator learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), Advances in Neural Information Processing Systems, 2022. URL https://openreview.net/forum?id=bF4eYy3LTR9.

Shekarpaz, S., Azizmalayeri, M., and Rohban, M. H. Piat: Physics informed adversarial training for solving partial differential equations. arXiv preprint arXiv:2207.06647, 2022.

Shi, Z., Zhang, H., Chang, K.-W., Huang, M., and Hsieh, C.-J. Robustness verification for transformers. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=BJxwPJHFwS.

Shin, Y., Darbon, J., and Karniadakis, G. E. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. arXiv preprint arXiv:2004.01806, 2020.

Sun, L., Gao, H., Pan, S., and Wang, J.-X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. Computer Methods in Applied Mechanics and Engineering, 361:112732, 2020.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., and Niepert, M. Pdebench: An extensive benchmark for scientific machine learning. Advances in Neural Information Processing Systems, 35: 1596–1611, 2022.

Wang, C., Li, S., He, D., and Wang, L. Is l^2 physics-informed loss always suitable for training physics-informed neural network? arXiv preprint arXiv:2206.02016, 2022a.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. arXiv preprint arXiv:2103.06624, 2021.

Wang, S., Yu, X., and Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. Journal of Computational Physics, 449:110768, 2022b.

Xu, K., Shi, Z., Zhang, H., Wang, Y., Chang, K.-W., Huang, M., Kailkhura, B., Lin, X., and Hsieh, C.-J. Automatic perturbation analysis for scalable certified robustness and beyond. Advances in Neural Information Processing Systems, 33:1129–1141, 2020a.

Xu, K., Zhang, H., Wang, S., Wang, Y., Jana, S., Lin, X., and Hsieh, C.-J. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. arXiv preprint arXiv:2011.13824, 2020b.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. Advances in neural information processing systems, 31, 2018.

A. Empirical relation of $|f|$ and $|u - u^*|$

One question that might arise from our certification procedure is the relationship between the PINN residual error $|f|$, and the solution error with respect to true solution u^*, $|u - u^*|$, across the domain. By definition, achieving a low $|f|$ implies u is a valid solution for the PDE, but there is no formal guarantee related $|u - u^*|$ within our framework.

Obtaining a bound on $|u - u^*|$ is typically a non-trivial task given u^* might not be unique, and does not necessarily exhibit an analytical solution and can only be computed using a numerical solver. And while some recent works perform this analysis for specific PDEs by exploiting their structure and/or smoothness properties (Mishra & Molinaro, 2022; Ryck & Mishra, 2022; Wang et al., 2022a), these methods typically suffer from scalability and bound tightness issues. As such, we perform an empirical analysis on Burgers' equation using a numerical, finite-difference solver to obtain $u^*(x)$ for sampled points x. We randomly sample 10^6 domain points S^0, and compute the maximum residual error, $\max_{x \in S^0} |f(x)|$, and the empirical maximum solution error, $\max_{x \in S^0} |u(x) - u^*(x)|$, for networks obtained at different epochs of the training process.

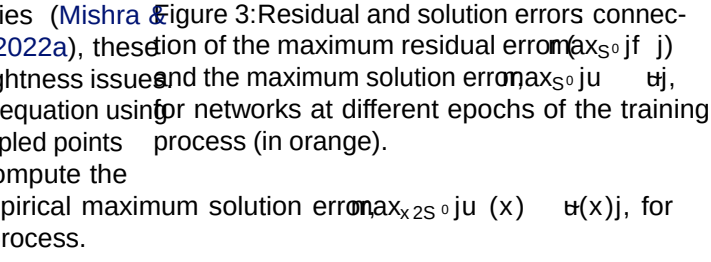


Figure 3: Residual and solution errors connection of the maximum residual error ($\max_{S^0} |f|$) and the maximum solution error, $\max_{S^0} |u - u^*|$, for networks at different epochs of the training process (in orange).

We report the results in Figure 3, with each point corresponding to an instance of a network. As expected, there is a correlation between these errors obtained using a numerical solver, suggesting a similar correlation holds for $|u - u^*|$.

B. On the importance of greedy input branching

A key factor in the success of CROWN in achieving tight bounds of the residual is the greedy input branching procedure from Algorithm 1. To illustrate the fact that a uniform sampling strategy would be significantly more computationally expensive, we plot in Figure 4 the relative density of branches (i.e., the percentage of branches per unit of input domain) in the case of Burgers' and Schrödinger's equations. As can be observed, there are clear imbalances at the level of the branching distribution – with areas away from relative optima u^* being relatively under sampled yet achieving tight bounds – showcasing the efficiency of our strategy.

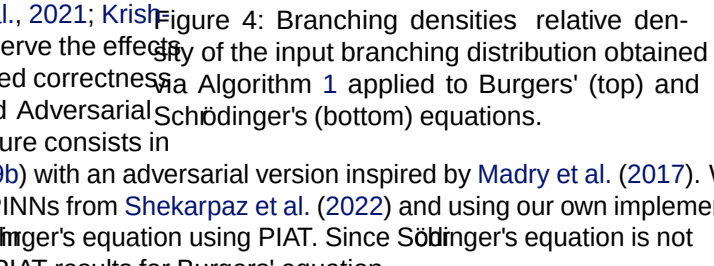


Figure 4: Branching densities relative density of the input branching distribution obtained via Algorithm 1 applied to Burgers' (top) and Schrödinger's (bottom) equations.

C. Reducing empirical and certified errors through Physics-Informed Adversarial Training

The goal of reducing the solution errors obtained by PINNs has been the research focus of several previous works (Kim et al., 2021; Krishnapriyan et al., 2021; Shekarpaz et al., 2022). To observe the effects of one of these different training schemes on the verified correctness, certification of PINNs, we consider Physics-informed Adversarial Training (PIAT) (Shekarpaz et al., 2022). The procedure consists in replacing the residual loss term from Raissi et al. (2019b) with an adversarial version inspired by Madry et al. (2017). While this procedure leads to improvements in the example PINNs from Shekarpaz et al. (2022) and using our own implementation in Burgers' equation, we were unable to stably train Schrödinger's equation using PIAT. Since Schrödinger's equation is not considered in Shekarpaz et al. (2022), we only show PIAT results for Burgers' equation.

We solve the inner optimization problem using 5 PGD steps (Madry et al., 2017), and $\epsilon = 0.05$ and a step size of 0.25. To improve convergence, we warm start PIAT training using a standard training solution after 6,000 L-BFGS iterations. The results in Table 2 show that as expected PIAT improves both empirical and certified residual bounds.

Table 2: PIAT on Burgers' equation. Monte Carlo sampled maximum values (10^8 samples in 0.21s) and upper bounds computed using @CROWN with N_b branchings for ① initial conditions ($t = 0, x \in D, N_b = 5$ k), ② boundary conditions ($t \in [0, T], x = \_1 \_ x = 1, N_b = 5$ k), and ③ residual norm ($t \in [0, T], x \in D, N_b = 125$k), for a PINN trained using PIAT from Shekarpaz et al. (2022).

		MC - max	@CROWN - u_b (time [s])
PIAT Burgers (Shekarpaz et al., 2022)	① $\|u(0, x) - u_0(x)\|^2$	$7.40 \cdot 10^{-6}$	$8.18 \cdot 10^{-6}$ (90.9)
	② $\|u(t, -1)\|^2$	$2.31 \cdot 10^{-7}$	$3.32 \cdot 10^{-7}$ (49.4)
	② $\|u(t, 1)\|^2$	$8.41 \cdot 10^{-8}$	$1.39 \cdot 10^{-7}$ (48.5)
	③ $\|f(x, t)\|^2$	$3.60 \cdot 10^{-3}$	$2.39 \cdot 10^{-2}$ ($2.8 \cdot 10^5$)

Certification convergence in PIAT vs. standard training. The regularization provided by adversarial training often leads to verification algorithms converging faster to tighter lower and upper bounds. We investigate whether this is the case with @CROWN's greedy branching strategy by comparing the relative convergence (i.e., the deviation between the upper bound and the empirical maximum, $|f^U - \max_{D'} |f||$) for the first 125k splits of PINNs trained in the standard and PIAT cases. The results presented in Figure 5 show that adversarial training leads to quicker convergence, requiring a lower number of branches to reach the same error when compared to standard. This suggests that our method, while already efficient, would benefit from smarter training strategies that lead to lower residual errors.

Figure 5: Certification Convergence. log-log plot of the relative convergence of @CROWN certification for a standard trained PINN (in blue) and PIAT (in orange).

D. Proofs of partial derivative computations

Within this section, we use $\partial_{x_i} f$ and $\frac{\partial f}{(\partial x_i)}$ interchangeably to refer to the jth partial derivative of a function $f : \mathbb{R}^n \to \mathbb{R}$ with respect to the i-component of its input x_i.

D.1. Lemma 1: computing $\partial_{x_i} u$

Lemma 1 (Computing $\partial_{x_i} u$). For $i \in \{1, \dots, d_0\}$, the partial derivative of u with respect to x_i can be computed recursively as $\partial_{x_i} u = W^{(L)} \partial_{x_i} z^{(L-1)}$ for:

$$\partial_{x_i} z^{(k)} = \partial_{z^{(k-1)}} z^{(k)} \partial_{x_i} z^{(k-1)}; \quad \partial_{x_i} z^{(0)} = e_i;$$

for $k \in \{1, \dots, L-1\}$, and where $\partial_{z^{(k-1)}} z^{(k)} = \text{diag}\left(\sigma'(y^{(k)})\right) W^{(k)}$.

Proof. Let us now derive $\partial_{x_i} u(x)$ for a given $i \in \{1, \dots, n_0\}$. Starting backwards from the last layer and applying the chain rule we obtain:

$$\partial_{x_i} u(x) = \frac{\partial y^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial z^{(L-2)}} \cdots \frac{\partial z^{(1)}}{\partial x} \frac{\partial x}{\partial x_i}$$

Given that $\partial_{x_i} x = e_i$ and $\frac{\partial y^{(L)}}{\partial z^{(L-1)}} = W^{(L)}$, all that's left to compute to obtain the full expression is $\frac{\partial z^{(k)}}{\partial z^{(k-1)}}$, $k \in \{L-1, \dots, 1\}$. Note that, for simplicity of the expressions, $z^{(0)} = x$. For every element $j \in \{1, \dots, d_k\}$ of $z^{(k)}$ denoted by $z_j^{(k)}$, we have:

$$\frac{\partial z_j^{(k)}}{\partial z^{(k-1)}} = \sigma'\left(W^{(k)}_{[j,:]} z^{(k-1)} + b_j^{(k)}\right) W^{(k)}_{[j,:]}$$

where $W^{(k)}_{j,:}$ denotes the j-th row of $W^{(k)}$, and $b_j^{(k)}$ the j-th element of b. Thus, the final expression can be obtained by stacking the columns of the previous expression to obtain the full Jacobian:

$$\frac{\partial z^{(k)}}{\partial z^{(k-1)}} = \text{diag}\left[\sigma'\left(W^{(k)} z^{(k-1)} + b^{(k)}\right)\right] W^{(k)}$$

This concludes the proof. □

D.2. Lemma 2: computing $\nabla^2_{x_i} u$

Lemma 2 (Expression for $\nabla^2_{x_i} u$ (x)). For $i \in \{1, \dots, d_0\}$, the second partial derivative of u with respect to x_i can be computed recursively as $\nabla^2_{x_i} u = W^{(L)} \nabla^2_{x_i} z^{(L-1)}$ where:

$$\nabla^2_{x_i} z^{(k)} = \nabla_{x_i z^{(k-1)}} z^{(k)} \nabla_{x_i} z^{(k-1)} + \nabla_{z^{(k-1)}} z^{(k)} \nabla^2_{x_i} z^{(k-1)},$$

and $\nabla^2_{x_i} z^{(0)} = 0$, for $k \in \{1, \dots, L-1\}$, with $\nabla_{x_i} z^{(k-1)}$ and $\nabla_{z^{(k-1)}} z^{(k)}$ as per in Lemma 1, and $\nabla_{x_i z^{(k-1)}} z^{(k)} = \operatorname{diag}\left(\sigma''(y^{(k)})\right) W^{(k)} \nabla_{x_i} z^{(k-1)} W^{(k)}$.

Proof. Given the result obtained in Appendix D.1, let us now derive $\nabla^2_{x_i} u$ (x) for a given $i \in \{1, \dots, d_0\}$. Starting backwards from the last layer of $\nabla_{x_i} u$ and applying the chain rule we obtain:

$$\nabla^2_{x_i} u = \frac{\partial}{\partial x_i} \left[\frac{\partial y^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial z^{(L-2)}} \cdots \frac{\partial z^{(1)}}{\partial x} \frac{\partial x}{\partial x_i} \right] = W^{(L)} \nabla^2_{x_i} z^{(L-1)}$$

Now the same can be applied to $\nabla^2_{x_i} z^{(L-1)}$, and in general to $\nabla^2_{x_i} z^{(k)}$ to obtain:

$$\nabla^2_{x_i} z^{(k)} = \frac{\partial}{\partial x_i} \left[\frac{\partial z^{(k)}}{\partial z^{(k-1)}} \nabla_{x_i} z^{(k-1)} \right] = \frac{\partial^2 z^{(k)}}{\partial x_i \partial z^{(k-1)}} \nabla_{x_i} z^{(k-1)} + \frac{\partial z^{(k)}}{\partial z^{(k-1)}} \nabla^2_{x_i} z^{(k-1)},$$

forming a recursion which can be taken until the first layer of u, i.e.,:

$$\nabla^2_{x_i} z^{(1)} = \frac{\partial}{\partial x_i} \left[\frac{\partial z^{(1)}}{\partial x} e_i \right] = \frac{\partial^2 z^{(1)}}{\partial x_i \partial x} e_i :$$

With the computation of $\nabla_{x_i} u$, both $\nabla_{x_i} z^{(k-1)}$ and $\frac{\partial z^{(k)}}{\partial z^{(k-1)}}$ are known. As such, the only missing pieces in the general recursion is the computation of $\frac{\partial^2 z^{(k)}}{\partial x_i \partial z^{(k-1)}}$. Recall from the previous section that $\frac{\partial z^{(k)}}{\partial z^{(k-1)}} = \operatorname{diag}\left(\sigma'\left(W^{(k)} z^{(k-1)} + b^{(k)}\right)\right) W^{(k)}$. As such:

$$\frac{\partial^2 z^{(k)}}{\partial x_i \partial z^{(k-1)}} = \frac{\partial}{\partial x_i} \left[\operatorname{diag}\left(\sigma'\left(W^{(k)} z^{(k-1)} + b^{(k)}\right)\right) W^{(k)} \right] :$$

Following the element-wise reasoning from above, we have that:

$$\begin{aligned} \frac{\partial^2 z_j^{(k)}}{\partial x_i \partial z^{(k-1)}} &= \sigma''\left(W^{(k)}_{j,:} z^{(k-1)} + b_j^{(k)}\right) \frac{\partial}{\partial x_i} \left[W^{(k)}_{j,:} z^{(k-1)} + b_j^{(k)} \right] W^{(k)}_{j,:} \\ &= \sigma''\left(W^{(k)}_{j,:} z^{(k-1)} + b_j^{(k)}\right) W^{(k)}_{j,:} \frac{\partial z^{(k-1)}}{\partial x_i} W^{(k)}_{j,:} \end{aligned}$$

Stacking as in the previous case, we obtain:

$$\frac{\partial^2 z^{(k)}}{\partial x_i \partial z^{(k-1)}} = \operatorname{diag}\left(\sigma''\left(W^{(k)} z^{(k-1)} + b^{(k)}\right) W^{(k)} \nabla_{x_i} z^{(k-1)}\right) W^{(k)};$$

completing the derivation of $\nabla^2_{x_i} u$ (x). □

D.3. Theorem 1: Formal Statement and Proof

Theorem 1 (@CROWN: linear lower and upper bounding of $\nabla_{x_i} u$). For every $j \in \{1, \dots, d_L\}$ there exist two functions $\nabla_{x_i} u^U_{;j}$ and $\nabla_{x_i} u^L_{;j}$ such that, $\forall x \in C$ it holds that $\nabla_{x_i} u^L_{;j} \leq \nabla_{x_i} u_{;j} \leq \nabla_{x_i} u^U_{;j}$, with:

$$\nabla_{x_i} u^U_{;j} = \Lambda^{(1),U}_{0;j,i} + \sum_{r=1}^{d_0} \Lambda^{(1),U}_{1;j,r} x + \Lambda^{(1),U}_{2;j,r}$$

$$\nabla_{x_i} u^L_{;j} = \Lambda^{(1),L}_{0;j,i} + \sum_{r=1}^{d_0} \Lambda^{(1),L}_{1;j,r} x + \Lambda^{(1),L}_{2;j,r}$$

where for $p \in \{0, 1, 2\}$, $\Omega^{(1),U}_{p,j,r}$ and $\Omega^{(1),L}_{p,j,r}$ are functions of $W^{(k)}$, $y^{(k),L}$, $y^{(k),U}$, $A^{(k),L}$, $A^{(k),U}$ $a^{(k),L}$, and $a^{(k),U}$, and can be computed using a recursive closed-form expression in $O(n)$ time.

Proof: Assume that through the computation of the previous bounds on the pre-activation layer outputs of $y^{(k)}$, are lower and upper bounded by linear functions de ned as $A^{(k),L}x + a^{(k),L} \leq y^{(k)} \leq A^{(k),U}x + a^{(k),U}$ and $y^{(k),L} \leq y^{(k)} \leq y^{(k),U}$ for $x \in C$.

Take the upper and lower bound functions for $\partial_i u$ as $\partial_i u^U$ and $\partial_i u^L$, respectively, and the upper and lower bound functions for $\partial_i z^{(k)}$ as $\partial_i z^{(k),U}$ and $\partial_i z^{(k),L}$, respectively. For the sake of simplicity of notation, we de ne $B^{(k),+} = I_{B^{(k)} \geq 0} B^{(k)}$ and $B^{(k),-} = I_{B^{(k)} < 0} B^{(k)}$.

Working backwards from $\partial_i u$, we apply the same idea from CROWN (Zhang et al., 2018):

$$\begin{aligned} \partial_i u^U &= W^{(L),+} \partial_i z^{(L-1),U} + W^{(L),-} \partial_i z^{(L-1),L} \\ \partial_i u^L &= W^{(L),+} \partial_i z^{(L-1),L} + W^{(L),-} \partial_i z^{(L-1),U} \end{aligned} \tag{6}$$

We continue to apply this backwards propagation rule to obtain $\partial_i z^{(L-1),U}$ and $\partial_i z^{(L-1),L}$. Recall that $\partial_i z^{(k)} = \partial_{z^{(k-1)}} z^{(k)} \partial_i z^{(k-1)}$, that is, for $j \in \{1, \dots, d_k\}$ we have $\partial_i z^{(k)}_j = \partial_{z^{(k-1)}} z^{(k)}_{j,:} \partial_i z^{(k-1)} = \sum_{n=1}^{d_{k-1}} \partial_{z^{(k-1)}} z^{(k)}_{j,n} \partial_i z^{(k-1)}_n$.

We resolve the bilinear dependencies of each $\partial_i z^{(k)}_j$ by relaxing it using a convex combination of the upper and lower bounds obtained by the McCormick envelopes of the product. Assuming that $\partial_{z^{(k-1)}} z^{(k),L}_{j,n} \leq \partial_{z^{(k-1)}} z^{(k)}_{j,n} \leq \partial_{z^{(k-1)}} z^{(k),U}_{j,n}$ and $\partial_i z^{(k-1),L}_n \leq \partial_i z^{(k-1)}_n \leq \partial_i z^{(k-1),U}_n$, we have that:

$$\begin{aligned} \partial_i z^{(k)}_j \geq \partial_i z^{(k),U}_j &= \sum_{n=1}^{d_{k-1}} \Lambda^{(k)}_{0,j,n} \partial_i z^{(k-1)}_n + \Lambda^{(k)}_{1,j,n} \partial_{z^{(k-1)}} z^{(k)}_{j,n} + \Omega^{(k)}_{2,j,n} \\ \partial_i z^{(k)}_j \leq \partial_i z^{(k),L}_j &= \sum_{n=1}^{d_{k-1}} \Lambda^{(k)}_{0,j,n} \partial_i z^{(k-1)}_n + \Lambda^{(k)}_{1,j,n} \partial_{z^{(k-1)}} z^{(k)}_{j,n} + \Omega^{(k)}_{2,j,n}; \end{aligned} \tag{7}$$

for:

$$\begin{aligned} \Lambda^{(k)}_{0,j,n} &= \Lambda^{(k)}_{j,n} \partial_{z^{(k-1)}} z^{(k),U}_{j,n} + (1 - \Lambda^{(k)}_{j,n}) \partial_{z^{(k-1)}} z^{(k),L}_{j,n} \\ \Lambda^{(k)}_{1,j,n} &= \Lambda^{(k)}_{j,n} \partial_i z^{(k-1),L}_n + (1 - \Lambda^{(k)}_{j,n}) \partial_i z^{(k-1),U}_n \\ \Omega^{(k)}_{2,j,n} &= \Lambda^{(k)}_{j,n} \partial_{z^{(k-1)}} z^{(k),U}_{j,n} \partial_i z^{(k-1),L}_n - (1 - \Lambda^{(k)}_{j,n}) \partial_{z^{(k-1)}} z^{(k),L}_{j,n} \partial_i z^{(k-1),U}_n \\ \Lambda^{(k)}_{0,j,n} &= \Lambda^{(k)}_{j,n} \partial_{z^{(k-1)}} z^{(k),L}_{j,n} + (1 - \Lambda^{(k)}_{j,n}) \partial_{z^{(k-1)}} z^{(k),U}_{j,n} \\ \Lambda^{(k)}_{1,j,n} &= \Lambda^{(k)}_{j,n} \partial_i z^{(k-1),L}_n + (1 - \Lambda^{(k)}_{j,n}) \partial_i z^{(k-1),U}_n \\ \Omega^{(k)}_{2,j,n} &= \Lambda^{(k)}_{j,n} \partial_{z^{(k-1)}} z^{(k),L}_{j,n} \partial_i z^{(k-1),L}_n - (1 - \Lambda^{(k)}_{j,n}) \partial_{z^{(k-1)}} z^{(k),U}_{j,n} \partial_i z^{(k-1),U}_n; \end{aligned}$$

where $\Lambda^{(k)}_{j,n}$ and $\Lambda^{(k)}_{j,n}$ are convex coef cients that can be set as hyperparameters, or optimized for as in CROWN (Xu et al., 2020b).

To continue the backward propagation, we now need to bound the components of $\partial_{z^{(k-1)}} z^{(k)}$. Recall from Lemma 1 that $\partial_{z^{(k-1)}} z^{(k)} = \text{diag}(\sigma'(y^{(k-1)})) W^{(k)}$, and $\partial_{z^{(k-1)}} z^{(k)}_{j,:} = \sigma'(y^{(k-1)}_j) W^{(k)}_{j,:}$ for $j \in \{1, \dots, d_k\}$.

Since $y^{(k),L}_j \leq y^{(k)}_j \leq y^{(k),U}_j$, we can obtain a linear upper and lower bound relaxation for $\sigma(y^{(k)}_j)$, such that

$\lambda_j^{(k);L}\, y_j^{(k)} + \mu_j^{(k);L} \quad 0 \quad y_j^{(k)} \quad \lambda_j^{(k);U}\, y_j^{(k)} + \mu_j^{(k);U}$. With this, we can proceed to bound $\frac{\partial z_{j,:}^{(k)}}{\partial z^{(k-1)}}$ as:

$$\begin{aligned}
\frac{\partial z_{j,:}^{(k)}}{\partial z^{(k-1)}} &\quad \underbrace{\lambda_j^{(k);U} W_{j,:}^{(k);+} + \lambda_j^{(k);L} W_{j,:}^{(k);-}}_{\{z_{0;j,:}^{(k)}\}} y_j^{(k)} + \underbrace{\lambda_j^{(k);U}\, \mu_j^{(k);U} W_{j,:}^{(k);+} + \lambda_j^{(k);L}\, \mu_j^{(k);L} W_{j,:}^{(k);-}}_{\{z_{1;j,:}^{(k)}\}} \\
\frac{\partial z_{j,:}^{(k)}}{\partial z^{(k-1)}} &\quad \underbrace{\lambda_j^{(k);L} W_{j,:}^{(k);+} + \lambda_j^{(k);U} W_{j,:}^{(k);-}}_{\{z_{0;j,:}^{(k)}\}} y_j^{(k)} + \underbrace{\lambda_j^{(k);L}\, \mu_j^{(k);L} W_{j,:}^{(k);+} + \lambda_j^{(k);U}\, \mu_j^{(k);U} W_{j,:}^{(k);-}}_{\{z_{1;j,:}^{(k)}\}}
\end{aligned} \tag{8}$$

At this point, one could continue the back-substitution process using the bounds from CROWN (Zhang et al., 2018). However, for the sake of efficiency, we use instead the pre-computed inequalities from propagating bounds through $A^{(k);U} x + a^{(k);U} \quad y^{(k)} \quad A^{(k);L} x + a^{(k);L}$. Substituting this in Equation 8, we obtain:

$$\begin{aligned}
\frac{\partial z_{j,:}^{(k);U}}{\partial z^{(k-1)}} &= \underbrace{\lambda_{0;j,:}^{(k);+} A_{j,:}^{(k);U} + \lambda_{0;j,:}^{(k);-} A_{j,:}^{(k);L}}_{\{z_{2;j,:}^{(k)}\}} x + \underbrace{\lambda_{0;j,:}^{(k);+} a_j^{(k);U} + \lambda_{0;j,:}^{(k);-} a_j^{(k);L} + \mu_{1;j,:}^{(k)}}_{\{z_{3;j,:}^{(k)}\}} \\
\frac{\partial z_{j,:}^{(k);L}}{\partial z^{(k-1)}} &= \underbrace{\lambda_{0;j,:}^{(k);+} A_{j,:}^{(k);L} + \lambda_{0;j,:}^{(k);-} A_{j,:}^{(k);U}}_{\{z_{2;j,:}^{(k)}\}} x + \underbrace{\lambda_{0;j,:}^{(k);+} a_j^{(k);L} + \lambda_{0;j,:}^{(k);-} a_j^{(k);U} + \mu_{1;j,:}^{(k)}}_{\{z_{3;j,:}^{(k)}\}}
\end{aligned} \tag{9}$$

In practice, we can use Equation 9 to compute the required $\frac{\partial z_{j,n}^{(k);L}}{\partial z^{(k-1)}}$ and $\frac{\partial z_{j,n}^{(k);U}}{\partial z^{(k-1)}}$ for the McCormick relaxation that leads to Equation 7. By back-substituting the result of Equation 9 in Equation 7, we obtain an expression for the upper and lower bounds of $\frac{\partial z_j^{(k)}}{\partial x_i}$ that only depends on $\frac{\partial z^{(k-1)}}{\partial x_i}$ and x:

$$\begin{aligned}
\frac{\partial z_j^{(k);U}}{\partial x_i} &= \sum_{n=1}^{d_{k-1}} \lambda_{0;j,n}^{(k)} \frac{\partial z_n^{(k-1)}}{\partial x_i} + \lambda_{3;j,n}^{(k)} x + \mu_{4;j,n}^{(k)} \\
\frac{\partial z_j^{(k);L}}{\partial x_i} &= \sum_{n=1}^{d_{k-1}} \lambda_{0;j,n}^{(k)} \frac{\partial z_n^{(k-1)}}{\partial x_i} + \lambda_{3;j,n}^{(k)} x + \mu_{4;j,n}^{(k)};
\end{aligned} \tag{10}$$

where:

$$\begin{aligned}
\lambda_{3;j,n}^{(k)} &= \lambda_{1;j,n}^{(k);+} \lambda_{2;j,n}^{(k)} + \lambda_{1;j,n}^{(k);-} \lambda_{2;j,n}^{(k)}; \qquad \mu_{4;j,n}^{(k)} = \lambda_{1;j,n}^{(k);+} \mu_{3;j,n}^{(k)} + \lambda_{1;j,n}^{(k);-} \mu_{3;j,n}^{(k)} + \mu_{2;j,n}^{(k)} \\
\lambda_{3;j,n}^{(k)} &= \lambda_{1;j,n}^{(k);+} \lambda_{2;j,n}^{(k)} + \lambda_{1;j,n}^{(k);-} \lambda_{2;j,n}^{(k)}; \qquad \mu_{4;j,n}^{(k)} = \lambda_{1;j,n}^{(k);+} \mu_{3;j,n}^{(k)} + \lambda_{1;j,n}^{(k);-} \mu_{3;j,n}^{(k)} + \mu_{2;j,n}^{(k)}
\end{aligned}$$

Given Equation 10, we now have a recursive expression for each of the blocks that compose the computation of $\frac{\partial u}{\partial x_i}$, which allows us to obtain a closed form expression for $\frac{\partial u^U}{\partial x_i}$ and $\frac{\partial u^L}{\partial x_i}$ by applying recursive back-substitution starting

with Equation 6. Let us begin by performing back-substitution to the result in Equation 10 for Layer 1:

$$\frac{\partial z_j^{(L-1),U}}{\partial z_i} = \sum_{n=1}^{d_{L-2}} \lambda_{0,j,n}^{(L-1)} \frac{\partial z_n^{(L-2)}}{\partial z_i} + \gamma_{3,j,n}^{(L-1)} x + \beta_{4,j,n}^{(L-1)} \tag{11}$$

$$= \sum_{n=1}^{d_{L-2}} \lambda_{0,j,n}^{(L-1)} \left(\sum_{r=1}^{d_{L-3}} \lambda_{0,n,r}^{(L-2)} \frac{\partial z_r^{(L-3)}}{\partial z_i} + \gamma_{3,n,r}^{(L-2)} x + \beta_{4,n,r}^{(L-2)} \right) + \gamma_{3,j,n}^{(L-1)} x + \beta_{4,j,n}^{(L-1)} \tag{12}$$

$$= \sum_{n=1}^{d_{L-2}} \lambda_{0,j,n}^{(L-1)} \left(\sum_{r=1}^{d_{L-3}} \lambda_{0,n,r}^{(L-2)} \frac{\partial z_r^{(L-3)}}{\partial z_i} \right) + \sum_{n=1}^{d_{L-2}} \lambda_{0,j,n}^{(L-1)} \left(\gamma_{3,n,r}^{(L-2)} x + \beta_{4,n,r}^{(L-2)} \right) + \gamma_{3,j,n}^{(L-1)} x + \beta_{4,j,n}^{(L-1)} \tag{13}$$

$$= \sum_{r=1}^{d_{L-3}} \left(\sum_{n=1}^{d_{L-2}} \lambda_{0,j,n}^{(L-1)} \lambda_{0,n,r}^{(L-2)} \right) \frac{\partial z_r^{(L-3)}}{\partial z_i} + \tag{14}$$

$$+ \sum_{n=1}^{d_{L-2}} \lambda_{0,j,n}^{(L-1)} \left(\gamma_{3,n,r}^{(L-2)} x + \beta_{4,n,r}^{(L-2)} + \frac{1}{d_{L-3}} \left(\gamma_{3,j,n}^{(L-1)} x + \beta_{4,j,n}^{(L-1)} \right) \right) \tag{15}$$

$$= \sum_{r=1}^{d_{L-3}} \lambda_{0,j,r}^{(L-2)} \frac{\partial z_r^{(L-3)}}{\partial z_i} + \sum_{n=1}^{d_{L-2}} \lambda_{0,j,n}^{(L-1)} \left(\gamma_{3,n,r}^{(L-2)} + \frac{1}{d_{L-3}} \gamma_{3,j,n}^{(L-1)} \right) x + \tag{16}$$

$$+ \sum_{n=1}^{d_{L-1}} \lambda_{0,j,n}^{(L-1)} \left(\beta_{4,n,r}^{(L-2)} + \frac{1}{d_{L-3}} \beta_{4,j,n}^{(L-1)} \right) \tag{17}$$

$$= \sum_{r=1}^{d_{L-3}} \lambda_{0,j,r}^{(L-2)} \frac{\partial z_r^{(L-3)}}{\partial z_i} + \gamma_{1,j,r}^{(L-2)} x + \beta_{2,j,r}^{(L-2)}; \tag{18}$$

where:

$$\lambda_{0,j,r}^{(L-2)} = \sum_{n=1}^{d_{L-2}} \lambda_{0,j,n}^{(L-1)} \lambda_{0,n,r}^{(L-2)}$$

$$\gamma_{1,j,r}^{(L-2)} = \sum_{n=1}^{d_{L-2}} \lambda_{0,j,n}^{(L-1)} \left(\gamma_{3,n,r}^{(L-2)} + \frac{1}{d_{L-2}} \gamma_{3,j,n}^{(L-1)} \right)$$

$$\beta_{2,j,r}^{(L-2)} = \sum_{n=1}^{d_{L-2}} \lambda_{0,j,n}^{(L-1)} \left(\beta_{4,n,r}^{(L-2)} + \frac{1}{d_{L-2}} \beta_{4,j,n}^{(L-1)} \right);$$

and:

$$\lambda_{p,n,:}^{(L-2)} = \begin{cases} \lambda_{p,n,:}^{(L-2)} & \text{if } \lambda_{0,j,n}^{(L-1)} \geq 0 \\ \mu_{p,n,:}^{(L-2)} & \text{if } \lambda_{0,j,n}^{(L-1)} < 0 \end{cases}, \quad p \in \{0, 3, 4\}$$

As in CROWN (Zhang et al., 2018), given we have put Equation 18 in the same form as Equation 11, we can now apply this argument recursively using the $\lambda^{(k)}$ and $\beta^{(k)}$ coefficients to obtain:

$$\frac{\partial z_j^{(L-1),U}}{\partial z_i} = \lambda_{0,j,i}^{(1)} + \sum_{r=1}^{d_0} \gamma_{1,j,r}^{(1)} x + \beta_{2,j,r}^{(1)};$$

where:

$$\begin{aligned}
\alpha^{(k-1)}_{0;j;r} &= \begin{cases} P^{(k)}_{0;j;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} P^{(k)}_{0;j;n} \, \alpha^{(k-1)}_{0;n;r} & \text{if } k \in \{2, \ldots, L-1\} \end{cases} \\
\alpha^{(k-1)}_{1;j;r} &= \begin{cases} P^{(k)}_{3;j;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} P^{(k)}_{0;j;n} \, \alpha^{(k-1)}_{3;n;r} + \frac{1}{d_{k-2}} \beta^{(k)}_{1;j;n} & \text{if } k \in \{2, \ldots, L-1\} \end{cases} \\
\alpha^{(k-1)}_{2;j;r} &= \begin{cases} P^{(k)}_{4;j;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} P^{(k)}_{0;j;n} \, \alpha^{(k-1)}_{4;n;r} + \frac{1}{d_{k-2}} \beta^{(k)}_{2;j;n} & \text{if } k \in \{2, \ldots, L-1\} \end{cases} ;
\end{aligned}$$

and:

$$\beta^{(k-1)}_{p;n;:} = \begin{cases} \alpha^{(k-1)}_{p;n;:} & \text{if } \beta^{(k)}_{0;j;n} \geq 0 \\ \alpha^{(k-1)}_{p;n;:} & \text{if } \beta^{(k)}_{0;j;n} < 0 \end{cases}, p \in \{0, 3, 4\}$$

And following the same recursive argument:

$$\partial_i z_j^{(L-1;L)} = \beta^{(1)}_{0;j;i} + \sum_{r=1}^{d_0} \beta^{(1)}_{1;j;r} \, x + \beta^{(1)}_{2;j;r} ;$$

where:

$$\begin{aligned}
\alpha^{(k-1)}_{0;j;r} &= \begin{cases} P^{(k)}_{0;j;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} P^{(k)}_{0;j;n} \, \alpha'^{(k-1)}_{0;n;r} & \text{if } k \in \{2, \ldots, L-1\} \end{cases} \\
\alpha^{(k-1)}_{1;j;r} &= \begin{cases} P^{(k)}_{3;j;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} P^{(k)}_{0;j;n} \, \alpha'^{(k-1)}_{3;n;r} + \frac{1}{d_{k-2}} \beta^{(k)}_{1;j;n} & \text{if } k \in \{2, \ldots, L-1\} \end{cases} \\
\alpha^{(k-1)}_{2;j;r} &= \begin{cases} P^{(k)}_{4;j;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} P^{(k)}_{0;j;n} \, \alpha'^{(k-1)}_{4;n;r} + \frac{1}{d_{k-2}} \beta^{(k)}_{2;j;n} & \text{if } k \in \{2, \ldots, L-1\} \end{cases} ;
\end{aligned}$$

and:

$$\alpha'^{(k-1)}_{p;n;:} = \begin{cases} \alpha^{(k-1)}_{p;n;:} & \text{if } \beta^{(k)}_{0;j;n} \geq 0 \\ \alpha^{(k-1)}_{p;n;:} & \text{if } \beta^{(k)}_{0;j;n} < 0 \end{cases}, p \in \{0, 3, 4\}$$

With these expressions, we can compute the required $\partial_i z_n^{(k-1;L)}$ and $\partial_i z_n^{(k-1;U)}$ which we assumed to be known to derive Equation 7.

Finally, by back-propagating the bounds starting from Equation 6, we get:

$$\begin{aligned}
\partial_i u^U_{;j} &= \sum_{n=1}^{d_1} W^{(L);+}_{j;n} \left(\sum_{r=1}^{d_2} \alpha^{(L-1)}_{0;n;r} \left(\partial_i z^{(L-2)}_{[r]} + \beta^{(L-1)}_{3;n;r} \, x + \beta^{(L-1)}_{4;n;r} \right) A + \right. \\
&\quad \left. + W^{(L);-}_{j;n} \left(\sum_{r=1}^{d_2} \alpha^{(L-1)}_{0;n;r} \left(\partial_i z^{(L-2)}_{[r]} + \beta^{(L-1)}_{3;n;r} \, x + \beta^{(L-1)}_{4;n;r} \right) \right) A \right) \\
&= \sum_{r=1}^{d_2} \left(\sum_{n=1}^{d_1} W^{(L);+}_{j;n} \, \alpha^{(L-1)}_{0;n;r} + W^{(L);-}_{j;n} \, \alpha'^{(L-1)}_{0;n;r} A \right) \partial_i z^{(L-2)}_{[r]} + \\
&\quad + \left(\sum_{n=1}^{d_1} W^{(L);+}_{j;n} \, \alpha^{(L-1)}_{3;n;r} + W^{(L);-}_{j;n} \, \alpha'^{(L-1)}_{3;n;r} A \right) x + \left(\sum_{n=1}^{d_1} W^{(L);+}_{j;n} \, \alpha^{(L-1)}_{4;n;r} + W^{(L);-}_{j;n} \, \alpha'^{(L-1)}_{4;n;r} A \right) \\
&= \sum_{r=1}^{d_2} \alpha^{(L-1);U}_{0;j;r} \, \partial_i z^{(L-2);U}_n + \beta^{(L-1);U}_{1;j;r} \, x + \beta^{(L-1);U}_{2;j;r} ;
\end{aligned}$$

where:

$$\alpha^{(L-1);U}_{0;j;r} = \sum_{n=1}^{d_{L-1}} W^{(L);+}_{j;n} \alpha^{(L-1)}_{0;n;r} + W^{(L);-}_{j;n} \alpha^{(L-1)}_{0;n;r}$$

$$\alpha^{(L-1);U}_{1;j;r} = \sum_{n=1}^{d_{L-1}} W^{(L);+}_{j;n} \alpha^{(L-1)}_{3;n;r} + W^{(L);-}_{j;n} \alpha^{(L-1)}_{3;n;r}$$

$$\alpha^{(L-1);U}_{2;j;r} = \sum_{n=1}^{d_{L-1}} W^{(L);+}_{j;n} \alpha^{(L-1)}_{4;n;r} + W^{(L);-}_{j;n} \alpha^{(L-1)}_{4;n;r}.$$

From this, using the same back-propagation logic as in the derivations of $\partial_i z_n^{(k-1);L}$ and $\partial_i z_n^{(k-1);U}$, we can obtain:

$$\partial_i u^U_{;j} = \alpha^{(1);U}_{0;j;i} + \sum_{r=1}^{d_0} \alpha^{(1);U}_{1;j;r} x + \alpha^{(1);U}_{2;j;r}; \tag{19}$$

where:

$$\alpha^{(k-1);U}_{0;j;r} = \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n} \alpha^{(k-1)}_{0;n;r} + W^{(k);-}_{j;n} \alpha^{(k-1)}_{0;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \alpha^{(k);U}_{0;j;n} \alpha^{(k-1)}_{0;n;r} & \text{if } k \in \{2; \dots; L-1\} \end{cases}$$

$$\alpha^{(k-1);U}_{1;j;r} = \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n} \alpha^{(k-1)}_{3;n;r} + W^{(k);-}_{j;n} \alpha^{(k-1)}_{3;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \alpha^{(k);U}_{0;j;n} \alpha^{(k-1)}_{3;n;r} + \frac{1}{d_{k-2}} \alpha^{(k);U}_{1;j;n} & \text{if } k \in \{2; \dots; L-1\} \end{cases}$$

$$\alpha^{(k-1);U}_{2;j;r} = \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n} \alpha^{(k-1)}_{4;n;r} + W^{(k);-}_{j;n} \alpha^{(k-1)}_{4;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \alpha^{(k);U}_{0;j;n} \alpha^{(k-1)}_{4;n;r} + \frac{1}{d_{k-2}} \alpha^{(k);U}_{2;j;n} & \text{if } k \in \{2; \dots; L-1\} \end{cases};$$

and:

$$\alpha^{(k-1)}_{p;n;:} = \begin{cases} \alpha^{(k-1)}_{p;n;:} & \text{if } \alpha^{(k);U}_{0;j;n} \geq 0 \\ \alpha^{(k-1)}_{p;n;:} & \text{if } \alpha^{(k);U}_{0;j;n} < 0 \end{cases}, p \in \{0; 3; 4\}$$

And similarly for the lower bound:

$$\partial_i u^L_{;j} = \alpha^{(1);L}_{0;j;i} + \sum_{r=1}^{d_0} \alpha^{(1);L}_{1;j;r} x + \alpha^{(1);L}_{2;j;r}; \tag{20}$$

where:

$$\alpha^{(k-1);L}_{0;j;r} = \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n} \alpha^{(k-1)}_{0;n;r} + W^{(k);-}_{j;n} \alpha^{(k-1)}_{0;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \alpha^{(k);L}_{0;j;n} \alpha^{(k-1)}_{0;n;r} & \text{if } k \in \{2; \dots; L-1\} \end{cases}$$

$$\alpha^{(k-1);L}_{1;j;r} = \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n} \alpha^{(k-1)}_{3;n;r} + W^{(k);-}_{j;n} \alpha^{(k-1)}_{3;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \alpha^{(k);L}_{0;j;n} \alpha^{(k-1)}_{3;n;r} + \frac{1}{d_{k-2}} \alpha^{(k);L}_{1;j;n} & \text{if } k \in \{2; \dots; L-1\} \end{cases}$$

$$\alpha^{(k-1);L}_{2;j;r} = \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n} \alpha^{(k-1)}_{4;n;r} + W^{(k);-}_{j;n} \alpha^{(k-1)}_{4;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \alpha^{(k);L}_{0;j;n} \alpha^{(k-1)}_{4;n;r} + \frac{1}{d_{k-2}} \alpha^{(k);L}_{2;j;n} & \text{if } k \in \{2; \dots; L-1\} \end{cases};$$

and:

$$\alpha^{(k-1)}_{p;n;:} = \begin{cases} \alpha^{(k-1)}_{p;n;:} & \text{if } \alpha^{(k);L}_{0;j;n} \geq 0 \\ \alpha^{(k-1)}_{p;n;:} & \text{if } \alpha^{(k);L}_{0;j;n} < 0 \end{cases}, p \in \{0; 3; 4\}:$$

D.4. Theorem 2 Formal Statement and Proof

Theorem 2 (@CROWN: linear lower and upper bounding @$_{\hat{x}_i^2} u$). Assume that through a previous computation of bounds on @$_{\hat{x}_i} u$, the components of that network required for @$_{\hat{x}_i^2} u$, i.e., @$_{\hat{x}_i} z^{(k-1)}$ and @$_{\hat{z}_{(k-1)}} z^{(k)}$, are lower and upper bounded by linear functions. In particular, $C^{(k);L} x + c^{(k);L} \leq$ @$_{\hat{x}_i} z^{(k-1)} \leq C^{(k);U} x + c^{(k);U}$ and $D^{(k);L} x + d^{(k);L} \leq$ @$_{\hat{z}_{(k-1)}} z^{(k)} \leq D^{(k);U} x + d^{(k);U}$.

For every $j \in \{1, \ldots, d_L\}$ there exist two functions @$_{\hat{x}_i^2} u^U_{;j}$ and @$_{\hat{x}_i^2} u^L_{;j}$ such that, $\forall x \in C$ it holds that @$_{\hat{x}_i^2} u^L_{;j} \leq$ @$_{\hat{x}_i^2} u_{;j} \leq$ @$_{\hat{x}_i^2} u^U_{;j}$. These functions can be written as:

$$\begin{aligned}
\text{@}_{\hat{x}_i^2} u^U_{;j} &= \Lambda^{(1);U}_{0;j;i} + \sum_{r=1}^{d_0} \Lambda^{(1);U}_{1;j;r} x + \Lambda^{(1);U}_{2;j;r} \\
\text{@}_{\hat{x}_i^2} u^L_{;j} &= \Lambda^{(1);L}_{0;j;i} + \sum_{r=1}^{d_0} \Lambda^{(1);L}_{1;j;r} x + \Lambda^{(1);L}_{2;j;r}
\end{aligned}$$

where for $p \in \{0, 1, 2\}$, $\Lambda^{(1);U}_{p;j;r}$ and $\Lambda^{(1);L}_{p;j;r}$ are functions of $W^{(k)}$, $y^{(k);L}$, $y^{(k);U}$, $A^{(k);L}$, $A^{(k);U}$ $a^{(k);L}$, $a^{(k);U}$, $C^{(k);L}$, $C^{(k);U}$ $c^{(k);L}$, $c^{(k);U}$, $D^{(k);L}$, $D^{(k);U}$ $d^{(k);L}$, and $d^{(k);U}$, and can be computed using a recursive closed-form expression in $O(L)$ time.

Proof: Assume that through the computation of the previous bounds on the pre-activation layer outputs of $y^{(k)}$, are lower and upper bounded by linear functions de ned as $A^{(k);L} x + a^{(k);L} \leq y^{(k)} \leq A^{(k);U} x + a^{(k);U}$ and $y^{(k);L} \leq y^{(k)} \leq y^{(k);U}$ for $x \in C$. Additionally, we consider also that through a previous computation of bounds on @$_{\hat{x}_i} u$, the components of that network required for @$_{\hat{x}_i^2} u$, i.e., @$_{\hat{x}_i} z^{(k-1)}$ and @$_{\hat{z}_{(k-1)}} z^{(k)}$ are lower and upper bounded by linear functions. In particular, $C^{(k);L} x + c^{(k);L} \leq$ @$_{\hat{x}_i} z^{(k-1)} \leq C^{(k);U} x + c^{(k);U}$ and $D^{(k);L} x + d^{(k);L} \leq$ @$_{\hat{z}_{(k-1)}} z^{(k)} \leq D^{(k);U} x + d^{(k);U}$.

Take the upper and lower bound functions for @$_{\hat{x}_i} u$ as @$_{\hat{x}_i^2} u^U$ and @$_{\hat{x}_i^2} u^L$, respectively, and the upper and lower bound functions for @$_{\hat{x}_i^2} z^{(k)}$ as @$_{\hat{x}_i^2} z^{(k);U}$ and @$_{\hat{x}_i^2} z^{(k);L}$, respectively. For the sake of simplicity of notation, we de ne $B^{(k);+} = I\{B^{(k)} \geq 0\} B^{(k)}$ and $B^{(k);-} = I\{B^{(k)} < 0\} B^{(k)}$.

Note that, unless explicitly mentioned otherwise, the non-network variables (denoted by Greek letters, as well as bold, capital and lowercase letters) used here have no relation to the ones from Appendix D.3

Starting backwards from @$_{\hat{x}_i^2} z^{(k)}$, we have that:

$$\text{@}_{\hat{x}_i^2} z^{(k)}_j = \sum_{n=1}^{d_{k-1}} \text{@}_{\hat{x}_i z^{(k-1)}} z^{(k)}_{j;n} \, \text{@}_{\hat{x}_i} z^{(k-1)}_n + \text{@}_{\hat{z}_{(k-1)}} z^{(k)}_{j;n} \, \text{@}_{\hat{x}_i^2} z^{(k-1)}_n .$$

Given the transitive property of the sum operator, we can bound @$_{\hat{x}_i^2} z^{(k)}_j$ by using a McCormick envelope around each of the multiplications. Assuming that for all $j \in \{1, \ldots, d_k\}, n \in \{1, \ldots, d_{k-1}\}$: @$_{\hat{x}_i z^{(k-1)}} z^{(k);L}_{j;n} \leq$ @$_{\hat{x}_i z^{(k-1)}} z^{(k)}_{j;n} \leq$ @$_{\hat{x}_i z^{(k-1)}} z^{(k);U}_{j;n}$, @$_{\hat{x}_i} z^{(k-1);L}_n \leq$ @$_{\hat{x}_i} z^{(k-1)}_n \leq$ @$_{\hat{x}_i} z^{(k-1);U}_n$, @$_{\hat{z}_{(k-1)}} z^{(k)}_{j;n} \leq$ @$_{\hat{z}_{(k-1)}} z^{(k)}_{j;n} \leq$ @$_{\hat{z}_{(k-1)}} z^{(k)}_{j;n}$, and @$_{\hat{x}_i^2} z^{(k-1);L}_n \leq$ @$_{\hat{x}_i^2} z^{(k-1)}_n \leq$ @$_{\hat{x}_i^2} z^{(k-1);U}_n$, we obtain:

$$\begin{aligned}
\text{@}_{\hat{x}_i^2} z^{(k)}_j \geq \text{@}_{\hat{x}_i^2} z^{(k);U}_j &= \sum_{n=1}^{d_{k-1}} \Omega^{(k)}_{0;j;n} \, \text{@}_{\hat{x}_i} z^{(k-1)}_n + \Omega^{(k)}_{1;j;n} \, \text{@}_{\hat{x}_i z^{(k-1)}} z^{(k)}_{j;n} + \Omega^{(k)}_{2;j;n} \, \text{@}_{\hat{x}_i^2} z^{(k-1)}_n + \Omega^{(k)}_{3;j;n} \, \text{@}_{\hat{z}_{(k-1)}} z^{(k)}_{j;n} + \Omega^{(k)}_{4;j;n} \\
\text{@}_{\hat{x}_i^2} z^{(k)}_j \leq \text{@}_{\hat{x}_i^2} z^{(k);L}_j &= \sum_{n=1}^{d_{k-1}} \Omega^{(k)}_{0;j;n} \, \text{@}_{\hat{x}_i} z^{(k-1)}_n + \Omega^{(k)}_{1;j;n} \, \text{@}_{\hat{x}_i z^{(k-1)}} z^{(k)}_{j;n} + \Omega^{(k)}_{2;j;n} \, \text{@}_{\hat{x}_i^2} z^{(k-1)}_n + \Omega^{(k)}_{3;j;n} \, \text{@}_{\hat{z}_{(k-1)}} z^{(k)}_{j;n} + \Omega^{(k)}_{4;j;n}
\end{aligned} \tag{21}$$

for:

$$\alpha^{(k)}_{0;j,n} = \alpha^{(k)}_{j,n} \frac{\partial}{\partial x_i} z^{(k-1)}_{j,n}^{(k);U} + (1 - \alpha^{(k)}_{j,n}) \frac{\partial}{\partial x_i} z^{(k-1)}_{j,n}^{(k);L} \qquad \alpha^{(k)}_{1;j,n} = \alpha^{(k)}_{j,n} \frac{\partial}{\partial x_i} z^{(k-1)}_{n}^{(k-1);L} + (1 - \alpha^{(k)}_{j,n}) \frac{\partial}{\partial x_i} z^{(k-1)}_{n}^{(k-1);U}$$

$$\alpha^{(k)}_{2;j,n} = \alpha^{(k)}_{j,n} \frac{\partial}{\partial z^{(k-1)}} z^{(k);U}_{j,n} + (1 - \alpha^{(k)}_{j,n}) \frac{\partial}{\partial z^{(k-1)}} z^{(k);L}_{j,n} \qquad \alpha^{(k)}_{3;j,n} = \alpha^{(k)}_{j,n} \frac{\partial}{\partial x_i^2} z^{(k-1)}_{n}^{(k-1);L} + (1 - \alpha^{(k)}_{j,n}) \frac{\partial}{\partial x_i^2} z^{(k-1)}_{n}^{(k-1);U}$$

$$\alpha^{(k)}_{4;j,n} = \alpha^{(k)}_{j,n} \frac{\partial}{\partial x_i} z^{(k-1)}_{j,n}^{(k);U} \frac{\partial}{\partial x_i} z^{(k-1)}_{n}^{(k-1);L} + (1 - \alpha^{(k)}_{j,n}) \frac{\partial}{\partial x_i} z^{(k-1)}_{j,n}^{(k);L} \frac{\partial}{\partial x_i} z^{(k-1)}_{n}^{(k-1);U} +$$
$$\alpha^{(k)}_{j,n} \frac{\partial}{\partial z^{(k-1)}} z^{(k);U}_{j,n} \frac{\partial}{\partial x_i} x_i z^{(k-1)}_{n}^{(k-1);L} + (1 - \alpha^{(k)}_{j,n}) \frac{\partial}{\partial z^{(k-1)}} z^{(k);L}_{j,n} \frac{\partial}{\partial x_i} x_i z^{(k-1)}_{n}^{(k-1);U}$$

$$\beta^{(k)}_{0;j,n} = \beta^{(k)}_{j,n} \frac{\partial}{\partial x_i} z^{(k-1)}_{j,n}^{(k);L} + (1 - \beta^{(k)}_{j,n}) \frac{\partial}{\partial x_i} z^{(k-1)}_{j,n}^{(k);U} \qquad \beta^{(k)}_{1;j,n} = \beta^{(k)}_{j,n} \frac{\partial}{\partial x_i} z^{(k-1)}_{n}^{(k-1);L} + (1 - \beta^{(k)}_{j,n}) \frac{\partial}{\partial x_i} z^{(k-1)}_{n}^{(k-1);U}$$

$$\beta^{(k)}_{2;j,n} = \beta^{(k)}_{j,n} \frac{\partial}{\partial z^{(k-1)}} z^{(k);L}_{j,n} + (1 - \beta^{(k)}_{j,n}) \frac{\partial}{\partial z^{(k-1)}} z^{(k);U}_{j,n} \qquad \beta^{(k)}_{3;j,n} = \beta^{(k)}_{j,n} \frac{\partial}{\partial x_i^2} z^{(k-1)}_{n}^{(k-1);L} + (1 - \beta^{(k)}_{j,n}) \frac{\partial}{\partial x_i^2} z^{(k-1)}_{n}^{(k-1);U}$$

$$\beta^{(k)}_{4;j,n} = \beta^{(k)}_{j,n} \frac{\partial}{\partial x_i} z^{(k-1)}_{j,n}^{(k);L} \frac{\partial}{\partial x_i} z^{(k-1)}_{n}^{(k-1);L} + (1 - \beta^{(k)}_{j,n}) \frac{\partial}{\partial x_i} z^{(k-1)}_{j,n}^{(k);U} \frac{\partial}{\partial x_i} z^{(k-1)}_{n}^{(k-1);U} +$$
$$\beta^{(k)}_{j,n} \frac{\partial}{\partial z^{(k-1)}} z^{(k);L}_{j,n} \frac{\partial}{\partial x_i^2} z^{(k-1)}_{n}^{(k-1);L} + (1 - \beta^{(k)}_{j,n}) \frac{\partial}{\partial z^{(k-1)}} z^{(k);U}_{j,n} \frac{\partial}{\partial x_i^2} z^{(k-1)}_{n}^{(k-1);U};$$

where $\alpha^{(k)}_{j,n}$, $\alpha^{(k)}_{j,n}$, $\beta^{(k)}_{j,n}$ and $\beta^{(k)}_{j,n}$ are convex coef cients that can be set as hyperparameters, or optimized for as CROWN (Xu et al., 2020b).

For the next step of the back-propagation process, we now need to bound $\frac{\partial}{\partial x_i} z^{(k-1)}_{n}$, $\frac{\partial}{\partial x_i} z^{(k-1)}_{j,n}$, and $\frac{\partial}{\partial z^{(k-1)}} z^{(k)}_{j,n}$, so as to eventually be able to write $\frac{\partial}{\partial x_i^2} z^{(k)}_{j}$ as a function of simply $\frac{\partial}{\partial x_i^2} z^{(k-1)}_{n}$ and x. As per our assumptions at the beginning of this section, for the sake of computational ef ciency we take $\frac{\partial}{\partial x_i^2} z^{(k-1)}_{n}$ and $\frac{\partial}{\partial z^{(k-1)}} z^{(k)}_{j,n}$ from the computation of the bounds of $\frac{\partial}{\partial x_i} u_{;j}$, and thus assume we have a linear upper and lower bound function on it. This leaves us with $\frac{\partial}{\partial x_i} z^{(k-1)}_{j,n}$ to bound as a linear function of x.

Note that, as per Lemma 2, $\frac{\partial}{\partial x_i} z^{(k-1)}_{j,n} = \frac{\partial^{00}}{} y^{(k)}_{j} \; W^{(k)}_{j,:} \frac{\partial}{\partial x_i} z^{(k-1)}_{n}$. Since $W^{(k)}_{j,:} \frac{\partial}{\partial x_i} z^{(k-1)}_{n} = \sum_{n=1}^{d_{k-1}} W^{(k)}_{j,n} \frac{\partial}{\partial x_i} z^{(k-1)}_{n}$, and $C^{(k);U}_{n,:} x + c^{(k);U}_{n} \geq \frac{\partial}{\partial x_i} z^{(k-1)}_{n} \geq C^{(k);L}_{n,:} x + c^{(k);L}_{n}$ (from the assumptions above), we can write:

$$W^{(k)}_{j,:} \frac{\partial}{\partial x_i} z^{(k-1)} \geq \underbrace{\left(\sum_{n=1}^{d_{k-1}} W^{(k);+}_{j,n} C^{(k);U}_{n,:} + W^{(k);-}_{j,n} C^{(k);L}_{n,:} \right)}_{E^{(k);U}_{j}} x + \underbrace{\left(\sum_{n=1}^{d_{k-1}} W^{(k);+}_{j,n} c^{(k);U}_{n} + W^{(k);-}_{j,n} c^{(k);L}_{n} \right)}_{e^{(k);U}_{j}}$$

$$W^{(k)}_{j,:} \frac{\partial}{\partial x_i} z^{(k-1)} \leq \underbrace{\left(\sum_{n=1}^{d_{k-1}} W^{(k);+}_{j,n} C^{(k);L}_{n,:} + W^{(k);-}_{j,n} C^{(k);U}_{n,:} \right)}_{E^{(k);L}_{j}} x + \underbrace{\left(\sum_{n=1}^{d_{k-1}} W^{(k);+}_{j,n} c^{(k);L}_{n} + W^{(k);-}_{j,n} c^{(k);U}_{n} \right)}_{e^{(k);L}_{j}};$$

We de ne $\beta^{(k);U}_{j} = \max_{x \in C} E^{(k);U}_{j} x + e^{(k);U}_{j}$ and $\beta^{(k);L}_{j} = \min_{x \in C} E^{(k);L}_{j} x + e^{(k);L}_{j}$. As with the rst derivative case, since $y^{(k);L}_{j} \leq y^{(k)}_{j} \leq y^{(k);U}_{j}$, we can obtain a linear upper and lower bound relaxation for $\frac{\partial^{00}}{} y^{(k)}_{j}$, such that $\beta^{(k);L}_{j} \frac{\partial}{\partial} y^{(k)}_{j} + \beta^{(k);L}_{j} \geq \frac{\partial^{00}}{} y^{(k)}_{j} \geq \beta^{(k);U}_{j} \frac{\partial}{\partial} y^{(k)}_{j} + \beta^{(k);U}_{j}$, as well as the values $\beta^{(k);L}_{j} \leq \frac{\partial^{00}}{} y^{(k)}_{j} \leq \beta^{(k);U}_{j}$. By considering the assumption that $A^{(k);U}_{j,:} x + a^{(k);U}_{j} \geq y^{(k)} \geq A^{(k);L}_{j,:} x + a^{(k);L}_{j}$, we can obtain:

$$\frac{\partial^{00}}{} y^{(k)}_{j} \geq \underbrace{\left(\beta^{(k);U;+}_{j} A^{(k);U}_{j,:} + \beta^{(k);U;-}_{j} A^{(k);L}_{j,:} \right)}_{H^{(k);U}_{j}} x + \underbrace{\left(\beta^{(k);U;+}_{j} a^{(k);U}_{j} + \beta^{(k);U;-}_{j} a^{(k);L}_{j} + \beta^{(k);U}_{j} \beta^{(k);U}_{j} \right)}_{h^{(k);U}_{j}}$$

$$\frac{\partial^{00}}{} y^{(k)}_{j} \leq \underbrace{\left(\beta^{(k);L;+}_{j} A^{(k);L}_{j,:} + \beta^{(k);L;-}_{j} A^{(k);U}_{j,:} \right)}_{H^{(k);L}_{j}} x + \underbrace{\left(\beta^{(k);L;+}_{j} a^{(k);L}_{j} + \beta^{(k);L;-}_{j} a^{(k);U}_{j} + \beta^{(k);L}_{j} \beta^{(k);L}_{j} \right)}_{h^{(k);L}_{j}};$$

This allows us to relax $-y_j^{(k)} \quad W_{j,:}^{(k)} \odot z^{(k-1)}$ using McCormick envelopes:

$$\begin{aligned} -y_j^{(k)} \quad W_{j,:}^{(k)} \odot z^{(k-1)} \quad & \underline{b}_{0,j}^{(k),U} \quad W_{j,:}^{(k)} \odot z^{(k-1)} + \overline{b}_{1,j}^{(k),U} \quad -y_j^{(k)} + \overline{b}_{2,j}^{(k),U} \\ -y_j^{(k)} \quad W_{j,:}^{(k)} \odot z^{(k-1)} \quad & \underline{b}_{0,j}^{(k),L} \quad W_{j,:}^{(k)} \odot z^{(k-1)} + \overline{b}_{1,j}^{(k),L} \quad -y_j^{(k)} + \overline{b}_{2,j}^{(k),L}; \end{aligned}$$

for:

$$\begin{aligned} \underline{b}_{0,j}^{(k),U} &= \underline{\alpha}_j^{(k)} \underline{b}_j^{(k),U} + \left(1 - \underline{\alpha}_j^{(k)}\right) \underline{b}_j^{(k),L} & \overline{b}_{1,j,n}^{(k),U} &= \underline{\alpha}_j^{(k)} \overline{W}_{j,n}^{(k)} + \left(1 - \underline{\alpha}_j^{(k)}\right) \overline{b}_j^{(k),U} \\ \overline{b}_{2,j}^{(k),U} &= \underline{\alpha}_j^{(k)} \overline{b}_j^{(k),U} - \underline{\alpha}_j^{(k)} \underline{b}_j^{(k),L} + \left(1 - \underline{\alpha}_j^{(k)}\right) & \end{aligned}$$

$$\begin{aligned} \underline{b}_{0,j}^{(k),L} &= \overline{\alpha}_j^{(k)} \underline{b}_j^{(k),L} + \left(1 - \overline{\alpha}_j^{(k)}\right) \overline{b}_j^{(k),U} & \overline{b}_{1,j}^{(k),L} &= \overline{\alpha}_j^{(k)} \overline{W}_{j,n}^{(k)} + \left(1 - \overline{\alpha}_j^{(k)}\right) \overline{b}_j^{(k),U} \\ \overline{b}_{2,j}^{(k),L} &= \overline{\alpha}_j^{(k)} \underline{b}_j^{(k),L} - \overline{\alpha}_j^{(k)} \overline{b}_j^{(k),U} + \left(1 - \overline{\alpha}_j^{(k)}\right); & \end{aligned}$$

where $\underline{\alpha}_j^{(k)}$ and $\overline{\alpha}_j^{(k)}$ are convex coefficients that can be set as hyperparameters, or optimized for as CROWN (Xu et al., 2020b). By replacing this multiplication in the expression from Lemma 2, we bound $\hat{z}_{j,n}^{(k)}$ as:

$$\begin{aligned} \hat{z}_{j,n}^{(k)} \quad & \underline{b}_{0,j,n}^{(k),U} \quad W_{j,:}^{(k)} \odot z^{(k-1)} + \overline{b}_{1,j,n}^{(k),U} \quad -y_j^{(k)} + \overline{b}_{2,j}^{(k),U} \\ \hat{z}_{j,n}^{(k)} \quad & \underline{b}_{0,j,n}^{(k),L} \quad W_{j,:}^{(k)} \odot z^{(k-1)} + \overline{b}_{1,j,n}^{(k),L} \quad -y_j^{(k)} + \overline{b}_{2,j}^{(k),L}; \end{aligned}$$

for:

$$\underline{b}_{i,j,n}^{(k),U} = \underline{b}_{i,j}^{(k),U} \overline{W}_{j,n}^{(k),+} + \overline{b}_{i,j}^{(k),L} \underline{W}_{j,n}^{(k),-}; \qquad \overline{b}_{i,j,n}^{(k),L} = \underline{b}_{i,j}^{(k),L} \overline{W}_{j,n}^{(k),+} + \overline{b}_{i,j}^{(k),U} \underline{W}_{j,n}^{(k),-} \qquad i \in \{0,1,2\}.$$

By replacing the lower and upper bounds for $-y_j^{(k)}$ and $W_{j,:}^{(k)} \odot z^{(k-1)}$ in the previous inequality, we obtain the expression:

$$\begin{aligned} \hat{z}_{j,n}^{(k)} \quad & \overline{M}_{j,n}^{(k),U} x + \overline{m}_{j,n}^{(k),U} \\ \hat{z}_{j,n}^{(k)} \quad & \overline{M}_{j,n}^{(k),L} x + \overline{m}_{j,n}^{(k),L}; \end{aligned}$$

for:

$$\begin{aligned} \overline{M}_{j,n}^{(k),U} &= \underline{E}_{0,j,n}^{(k),+} \overline{E}_j^{(k),U} + \overline{E}_{0,j,n}^{(k),-} \underline{E}_j^{(k),L} + \underline{H}_{1,j,n}^{(k),+} \overline{H}_j^{(k),U} + \overline{H}_{1,j,n}^{(k),-} \underline{H}_j^{(k),L} \\ \overline{m}_{j,n}^{(k),U} &= \underline{e}_{0,j,n}^{(k),+} \overline{e}_j^{(k),U} + \overline{e}_{0,j,n}^{(k),-} \underline{e}_j^{(k),L} + \underline{h}_{1,j,n}^{(k),+} \overline{h}_j^{(k),U} + \overline{h}_{1,j,n}^{(k),-} \underline{h}_j^{(k),L} + \overline{b}_{2,j,n}^{(k),U} \\ \overline{M}_{j,n}^{(k),L} &= \underline{E}_{0,j,n}^{(k),+} \underline{E}_j^{(k),L} + \overline{E}_{0,j,n}^{(k),-} \overline{E}_j^{(k),U} + \underline{H}_{1,j,n}^{(k),+} \underline{H}_j^{(k),L} + \overline{H}_{1,j,n}^{(k),-} \overline{H}_j^{(k),U} \\ \overline{m}_{j,n}^{(k),L} &= \underline{e}_{0,j,n}^{(k),+} \underline{e}_j^{(k),L} + \overline{e}_{0,j,n}^{(k),-} \overline{e}_j^{(k),U} + \underline{h}_{1,j,n}^{(k),+} \underline{h}_j^{(k),L} + \overline{h}_{1,j,n}^{(k),-} \overline{h}_j^{(k),U} + \underline{b}_{2,j,n}^{(k),L}; \end{aligned}$$

Finally in the derivation of $\hat{z}_j^{(k)}$ as a function of x and $\hat{z}^{(k-1)}$, we just have to replace all the quantities in Equation 21 (recalling from the assumptions that $\underline{c}^{(k),U} x + \underline{c}^{(k),U} \odot z^{(k-1)} \quad \overline{C}^{(k),L} x + \overline{c}^{(k),L}$ and $\underline{D}^{(k),U} x + \underline{d}^{(k),U} \quad \overline{D}^{(k),L} x + \overline{d}^{(k),L}$) to obtain:

$$\begin{aligned} \hat{z}_j^{(k)} \quad \hat{z}_j^{(k),U} &= \sum_{n=1}^{d_{k-1}} \overline{A}_{2,j,n}^{(k)} \hat{z}_n^{(k-1)} + \overline{A}_{5,j,n}^{(k)} x + \overline{A}_{6,j,n}^{(k)} \\ \hat{z}_j^{(k)} \quad \hat{z}_j^{(k),L} &= \sum_{n=1}^{d_{k-1}} \underline{A}_{2,j,n}^{(k)} \hat{z}_n^{(k-1)} + \underline{A}_{5,j,n}^{(k)} x + \underline{A}_{6,j,n}^{(k)}; \end{aligned} \tag{22}$$

where:

$$\begin{aligned}
\overset{(k)}{5;j;n} &= \overset{(k);+}{0;j;n} C_n^{(k);U} + \overset{(k);-}{0;j;n} C_n^{(k);L} + \overset{(k);+}{1;j;n} M_{j;n}^{(k);U} + \overset{(k);-}{1;j;n} M_{j;n}^{(k);L} + \overset{(k);+}{3;j;n} D_{j;n}^{(k);U} + \overset{(k);-}{3;j;n} D_{j;n}^{(k);L} \\
\overset{(k)}{6;j;n} &= \overset{(k);+}{0;j;n} c_n^{(k);U} + \overset{(k);-}{0;j;n} c_n^{(k);L} + \overset{(k);+}{1;j;n} m_{j;n}^{(k);U} + \overset{(k);-}{1;j;n} m_{j;n}^{(k);L} + \overset{(k);+}{3;j;n} d_{j;n}^{(k);U} + \overset{(k);-}{3;j;n} d_{j;n}^{(k);L} + \overset{(k)}{4;j;n} \\
\overset{(k)}{5;j;n} &= \overset{(k);+}{0;j;n} C_n^{(k);L} + \overset{(k);-}{0;j;n} C_n^{(k);U} + \overset{(k);+}{1;j;n} M_{j;n}^{(k);L} + \overset{(k);-}{1;j;n} M_{j;n}^{(k);U} + \overset{(k);+}{3;j;n} D_{j;n}^{(k);L} + \overset{(k);-}{3;j;n} D_{j;n}^{(k);U} \\
\overset{(k)}{6;j;n} &= \overset{(k);+}{0;j;n} c_n^{(k);L} + \overset{(k);-}{0;j;n} c_n^{(k);U} + \overset{(k);+}{1;j;n} m_{j;n}^{(k);L} + \overset{(k);-}{1;j;n} m_{j;n}^{(k);U} + \overset{(k);+}{3;j;n} d_{j;n}^{(k);L} + \overset{(k);-}{3;j;n} d_{j;n}^{(k);U} + \overset{(k)}{4;j;n}
\end{aligned}$$

This forms a recursion of exactly the same form as Equation 10 from Appendix D.3, where only the coefficients of $@_{x_i} z_n^{(k\ 1)}$ and$x$ are different ($\overset{(k)}{0;j;n}$ in this case is referred by $\overset{(k)}{2;j;n}$, $\overset{(k)}{3;j;n}$ by $\overset{(k)}{5;j;n}$, and $\overset{(k)}{4;j;n}$ by $\overset{(k)}{6;j;n}$, and similarly for the values). This yields:

$$@_{i\ x_i} z_j^{(L\ \ 1);U} = \overset{(1)\ ;U}{0;j;i} + \sum_{r=1}^{d_0} \overset{(1)\ ;U}{1;j;r} x + \overset{(1)\ ;U}{2;j;r};$$

where:

$$\begin{aligned}
\overset{(k\ \ 1);U}{0;j;r} &= \begin{cases} P^{\overset{(k)}{2;n;r}}_{n=1} \overset{(k);U}{0;j;n} \overset{(k\ \ 1);U}{2;n;r} & \text{if } k = L \\ \text{if } k \in \{2; : : : ; L\ \ 1\} \end{cases} \\
\overset{(k\ \ 1);U}{1;j;r} &= \begin{cases} P^{\overset{(k)}{5;n;r}}_{n=1} \overset{(k);U}{0;j;n} \overset{(k\ \ 1);U}{5;n;r} + \frac{1}{d_{k\ \ 2}} \overset{(k);U}{1;j;n} & \text{if } k = L \\ \text{if } k \in \{2; : : : ; L\ \ 1\} \end{cases} \\
\overset{(k\ \ 1);U}{2;j;r} &= \begin{cases} P^{\overset{(k)}{6;n;r}}_{n=1} \overset{(k);U}{0;j;n} \overset{(k\ \ 1);U}{6;n;r} + \frac{1}{d_{k\ \ 2}} \overset{(k);U}{2;j;n} & \text{if } k = L \\ \text{if } k \in \{2; : : : ; L\ \ 1\} \end{cases}
\end{aligned};$$

and:

$$\overset{(k\ \ 1);U}{p;n;:} = \begin{cases} \overset{(k\ \ 1)}{p;n;:} & \text{if } \overset{(k);U}{0;j;n} \geq 0 \\ \overset{(k\ \ 1)}{p;n;:} & \text{if } \overset{(k);U}{0;j;n} < 0 \end{cases}, p \in \{2; 5; 6\}:$$

And following the same argument:

$$@_{i} z_j^{(L\ \ 1);L} = \overset{(1)\ ;L}{0;j;i} + \sum_{r=1}^{d_0} \overset{(1)\ ;L}{1;j;r} x + \overset{(1)\ ;L}{2;j;r};$$

where:

$$\begin{aligned}
\overset{(k\ \ 1);L}{0;j;r} &= \begin{cases} P^{\overset{(k)}{2;n;r}}_{n=1} \overset{(k);L}{0;j;n} \overset{(k\ \ 1);L}{2;n;r} & \text{if } k = L \\ \text{if } k \in \{2; : : : ; L\ \ 1\} \end{cases} \\
\overset{(k\ \ 1);L}{1;j;r} &= \begin{cases} P^{\overset{(k)}{5;n;r}}_{n=1} \overset{(k);L}{0;j;n} \overset{(k\ \ 1);L}{5;n;r} + \frac{1}{d_{k\ \ 2}} \overset{(k);L}{1;j;n} & \text{if } k = L \\ \text{if } k \in \{2; : : : ; L\ \ 1\} \end{cases} \\
\overset{(k\ \ 1);L}{2;j;r} &= \begin{cases} P^{\overset{(k)}{6;n;r}}_{n=1} \overset{(k);L}{0;j;n} \overset{(k\ \ 1);L}{6;n;r} + \frac{1}{d_{k\ \ 2}} \overset{(k);L}{2;j;n} & \text{if } k = L \\ \text{if } k \in \{2; : : : ; L\ \ 1\} \end{cases}
\end{aligned};$$

and:

$$\overset{(k\ \ 1);L}{p;n;:} = \begin{cases} \overset{(k\ \ 1)}{p;n;:} & \text{if } \overset{(k);L}{0;j;n} \geq 0 \\ \overset{(k\ \ 1)}{p;n;:} & \text{if } \overset{(k);L}{0;j;n} < 0 \end{cases}, p \in \{2; 5; 6\}$$

With these expressions, we can compute the required $@_{x_i} z_n^{(k\ \ 1);L}$ and $@_{x_i^2} z_n^{(k\ \ 1);U}$ which we assumed to be known to derive Equation 21.

Finally, with the exact same argument as in Appendix D.3, we obtain:

$$@_{x_i} u^{U}_{;j} = {}^{(1);U}_{0;j;i} + \sum_{r=1}^{d_0} {}^{(1);U}_{1;j;r}\, x + {}^{(1);U}_{2;j;r};$$

where:

$$
\begin{aligned}
{}^{(k-1);U}_{0;j;r} &= \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n}\, {}^{(k-1)}_{2;n;r} + W^{(k);-}_{j;n}\, {}^{(k-1)}_{2;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} {}^{(k);U}_{0;j;n}\, {}^{(k-1);U}_{2;n;r} & \text{if } k \in \{2;\dots;L-1\} \end{cases} \\
{}^{(k-1);U}_{1;j;r} &= \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n}\, {}^{(k-1)}_{5;n;r} + W^{(k);-}_{j;n}\, {}^{(k-1)}_{5;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} {}^{(k);U}_{0;j;n}\, {}^{(k-1);U}_{5;n;r} + \frac{1}{d_{k-2}}\, {}^{(k);U}_{1;j;n} & \text{if } k \in \{2;\dots;L-1\} \end{cases} \\
{}^{(k-1);U}_{2;j;r} &= \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n}\, {}^{(k-1)}_{6;n;r} + W^{(k);-}_{j;n}\, {}^{(k-1)}_{6;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} {}^{(k);U}_{0;j;n}\, {}^{(k-1);U}_{6;n;r} + \frac{1}{d_{k-2}}\, {}^{(k);U}_{2;j;n} & \text{if } k \in \{2;\dots;L-1\} \end{cases};
\end{aligned}
$$

and:

$${}^{(k-1)}_{p;n;:} = \begin{cases} {}^{(k-1)}_{p;n;:} & \text{if } {}^{(k);U}_{0;j;n} \geq 0 \\ {}^{(k-1)}_{p;n;:} & \text{if } {}^{(k);U}_{0;j;n} < 0 \end{cases}, p \in \{2;5;6\}.$$

And similarly for the lower bound:

$$@_{x_i} u^{L}_{;j} = {}^{(1);L}_{0;j;i} + \sum_{r=1}^{d_0} {}^{(1);L}_{1;j;r}\, x + {}^{(1);L}_{2;j;r};$$

where:

$$
\begin{aligned}
{}^{(k-1);L}_{0;j;r} &= \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n}\, {}^{(k-1)}_{2;n;r} + W^{(k);-}_{j;n}\, {}^{(k-1)}_{2;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} {}^{(k);L}_{0;j;n}\, {}^{(k-1);L}_{2;n;r} & \text{if } k \in \{2;\dots;L-1\} \end{cases} \\
{}^{(k-1);L}_{1;j;r} &= \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n}\, {}^{(k-1)}_{5;n;r} + W^{(k);-}_{j;n}\, {}^{(k-1)}_{5;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} {}^{(k);L}_{0;j;n}\, {}^{(k-1);L}_{5;n;r} + \frac{1}{d_{k-2}}\, {}^{(k);L}_{1;j;n} & \text{if } k \in \{2;\dots;L-1\} \end{cases} \\
{}^{(k-1);L}_{2;j;r} &= \begin{cases} \sum_{n=1}^{d_{k-1}} W^{(k);+}_{j;n}\, {}^{(k-1)}_{6;n;r} + W^{(k);-}_{j;n}\, {}^{(k-1)}_{6;n;r} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} {}^{(k);L}_{0;j;n}\, {}^{(k-1);L}_{6;n;r} + \frac{1}{d_{k-2}}\, {}^{(k);L}_{2;j;n} & \text{if } k \in \{2;\dots;L-1\} \end{cases};
\end{aligned}
$$

and:

$${}^{(k-1);L}_{p;n;:} = \begin{cases} {}^{(k-1)}_{p;n;:} & \text{if } {}^{(k);L}_{0;j;n} \geq 0 \\ {}^{(k-1)}_{p;n;:} & \text{if } {}^{(k);L}_{0;j;n} < 0 \end{cases}, p \in \{2;5;6\}.$$

D.5. Formulation and proof of closed-form global bounds on $@_{x_i} u$

Lemma 3 (Closed-form global bounds on $@_{x_i} u$). For every $j \in \{1;\dots;d_L\}$ there exist two values ${}^{U}_{j} \in \mathbb{R}$ and ${}^{L}_{j} \in \mathbb{R}$, such that $\forall x \in \mathcal{C} = \{x \in \mathbb{R}^{d_0} : x^L \leq x \leq x^U\}$ it holds that ${}^{L}_{j} \leq @_{x_i} u_{;j} \leq {}^{U}_{j}$, with:

$${}^{U}_{j} = B^{U;+} x^U + B^{U;-} x^L + {}^{(1)}_{0;j;i} + \sum_{r=1}^{d_0} {}^{(1)}_{2;j;r}$$

$${}^{L}_{j} = B^{L;+} x^L + B^{L;-} x^U + {}^{(1)}_{0;j;i} + \sum_{r=1}^{d_0} {}^{(1)}_{2;j;r};$$

where $B^U = \sum_{r=1}^{d_0} {}^{(1)}_{1;j;r}$, $B^L = \sum_{r=1}^{d_0} {}^{(1)}_{1;j;r}$, and $B^{;+} = \mathbb{I}(B \geq 0) \odot B$ and $B^{;-} = \mathbb{I}(B < 0) \odot B$.

Table 3: Relaxing $\sigma'(y) = 1 - \tanh^2(y)$: linear upper and lower bounds for a given l_b and u_b.

l_b	u_b	U	U	L	L
R_1 R_3	R_1 R_3	$(\sigma(u_b) - \sigma(l_b))/(u_b - l_b)$	$\sigma(l_b) = {}^{U}$ l_b	$\sigma'(d), d \in [l_b, u_b]$	$\sigma(d) = {}^{L}$ d
R_2	R_2	$\sigma'(d), d \in [l_b, u_b]$	$\sigma(d) = {}^{U}$ d	$(\sigma(u_b) - \sigma(l_b))/(u_b - l_b)$	$\sigma(l_b) = {}^{L}$ l_b
R_1	R_2	$\sigma'(d_1)$, ${}_{y_1;u_b}(\sigma'; l_b; d_1) = 0$	$\sigma(l_b) = {}^{U}$ l_b	$\sigma'(d_2)$, ${}_{l_b;y_1}(\sigma'; u_b; d_2) = 0$	$\sigma(u_b) = {}^{L}$ u_b
R_2	R_3	$\sigma'(d_1)$, ${}_{l_b;y_2}(\sigma'; u_b; d_1) = 0$	$\sigma(u_b) = {}^{U}$ u_b	$\sigma'(d_2)$, ${}_{y_2;u_b}(\sigma'; l_b; d_2) = 0$	$\sigma(l_b) = {}^{L}$ l_b
R_1	R_3	$\sigma'(d_1) + (1 -)\sigma'(d_2)$, ${}_{l_b;0}(\sigma'; l_b; d_1) = 0$, ${}_{0;u_b}(\sigma'; u_b; d_2) = 0$	${}^{U}_{1} + (1 -){}^{U}_{2}$, ${}^{U}_{1} = \sigma(l_b) - \sigma'(d_1)\, l_b$, ${}^{U}_{2} = \sigma(u_b) - \sigma'(d_2)\, u_b$	$\sigma'(d_3)$, $l_b \geq u_b$ $\sigma'(d_4)$, $l_b < u_b$, ${}_{l_b;y_1}(\sigma'; u_b; d_3) = 0$, ${}_{y_2;u_b}(\sigma'; l_b; d_4) = 0$	$\begin{cases} \frac{\sigma(u_b)}{\sigma'(d_3)}\, u_b, & l_b \geq u_b \\ \frac{\sigma(l_b)}{\sigma'(d_4)}\, l_b, & l_b < u_b \end{cases}$

Proof. Take a function $f : \mathbb{R}^{d_0} \to \mathbb{R}$ defined as $f(x) = v^\top x + c$ for $v \in \mathbb{R}^{d_0}$ and $c \in \mathbb{R}$, as well as a domain $C = \{x \in \mathbb{R}^{d_0} : x^L \leq x \leq x^U\}$. Given the perpendicularity of the constraints in C, by separating each component of v we obtain:

$$\max_{x \in C} f(x) = (v^+)^\top x^U + (v^-)^\top x^L + c; \qquad \min_{x \in C} f(x) = (v^+)^\top x^L + (v^-)^\top x^U + c;$$

where $v^+ = \mathbb{I}(v \geq 0) \odot v$ and $v^- = \mathbb{I}(v < 0) \odot v$. □

E. Correctness Certification for PINNs with tanh activations

@CROWN allows one to compute lower and upper bounds on the outputs of $\partial_{x_i} u$ and f as long as we can obtain linear bounds for u's activations, σ, $\partial_{x_i} u$'s activations, σ', and $\partial^2_{x_i} u$'s activations, σ'', assuming previously computed bounds on the input of those activations. In this section we explore how to compute those bounds when σ is $\tanh$ activations.

Throughout, we assume the activation's input y to σ is lower bounded by l_b and upper bounded by u_b (i.e., $l_b \leq y \leq u_b$), and define the upper bound line as $h^U(y) = {}^{U}(y + {}^{U})$, and the lower bound line as $h^L(y) = {}^{L}(y + {}^{L})$. For the sake of brevity, we define for a function $h : \mathbb{R} \to \mathbb{R}$, and points $p, d \in \mathbb{R}$ the function $\Delta(h; p; d) = (h(p) - h(d))/(p - d) - h'(d)$. This is useful as for a given h and p, if there exists a $d \in [d_l, d_u]$, such that $\min_{d_l \leq d \leq d_u} \Delta(h; p; d) = 0$, then $h'(d)$ is the slope of a tangent line to h that passes through p and d.

Bounding $\sigma(y) = \tanh(y)$. We follow the bounds provided in CROWN (Zhang et al., 2018), by observing that $\tanh$ is a convex function for $y < 0$ and concave for $y > 0$. For $l_b \leq u_b \leq 0$ we let h^U be the line that connects l_b and u_b, and for an arbitrary $d \in [l_b, u_b]$ we let h^L be the tangent line at that point. Similarly, for $0 \leq l_b \leq u_b$ we let h^L be the line that connects l_b and u_b, and for an arbitrary $d \in [l_b, u_b]$ we let h^U be the tangent line at that point. For the last case where $l_b \leq 0 \leq u_b$, we let h^U be the tangent line at $d_1 \leq 0$ that passes through $(l_b, \sigma(l_b))$, and h^L be the tangent line at $d_2 \geq 0$ that passes through $(u_b, \sigma(u_b))$. Given these bounds were given in Zhang et al. (2018), we omit visual representations of them.

Bounding $\sigma'(y) = 1 - \tanh^2(y)$. The derivative of $\tanh(y)$, $1 - \tanh^2(y)$, is a more complicated function. By inspecting it's derivative, $\sigma''(y) = -2\tanh(y)(1 - \tanh^2(y))$, we conclude that there are two inflection points $y_1 = \max \sigma''(y)$ and $y_2 = \min \sigma''(y)$, leading to three different regions $y \in]-\infty; y_1]$ (R_1, the first convex region) $y \in]y_1; y_2]$ (R_2, the concave region), and $y \in]y_2; +\infty[$ (R_3, the second convex region). As a result, there are 6 combinations for the location of l_b and u_b which must be resolved.

The first two cases are the straightforward. If $l_b \in R_1$ and $u_b \in R_1$ or $l_b \in R_3$ and $u_b \in R_3$, i.e., if both ends are in the same convex region, then we use the same relaxation as in the bounding of the convex region. h^U is the line that connects l_b and u_b, while h^L is a tangent line at a point $d \in [l_b, u_b]$. Similarly for the case where $l_b \in R_2$ and $u_b \in R_2$, we take the solution from the $\tanh$ concave side and use h^L to be the line that connects l_b and u_b, and h^U to be the tangent line at a point $d \in [l_b, u_b]$. The next case is $l_b \in R_1$ and $u_b \in R_2$, i.e., l_b in the first convex region and u_b in the concave one. In

(a) $l_b \in R_1$ and $u_b \in R_2$ (b) $l_b \in R_2$ and $u_b \in R_3$ (c) $l_b \in R_1$ and $u_b \in R_3$

Figure 6: Relaxing $\sigma'(y) = 1 - \tanh^2(y)$: examples of the linear relaxations of σ' for different sets of l_b and u_b.

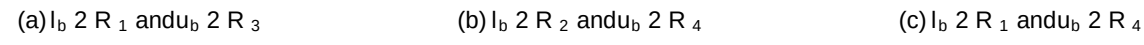
Table 4: Relaxing $\sigma''(y) = -2\tanh(y)(1 - \tanh^2(y))$: linear upper and lower bounds for a given l_b and u_b.

l_b	u_b	U	U	L	L
R_1 R_3	R_1 R_3	$(\sigma''(u_b) - \sigma''(l_b))/(u_b - l_b)$	$\sigma''(l_b) = U \cdot l_b$	$\sigma'''(d), d \in [l_b, u_b]$	$\sigma''(d) = L \cdot d$
R_2 R_4	R_2 R_4	$\sigma'''(d), d \in [l_b, u_b]$	$\sigma''(d) = U \cdot d$	$(\sigma''(u_b) - \sigma''(l_b))/(u_b - l_b)$	$\sigma''(l_b) = L \cdot l_b$
R_1	R_2	$\sigma'''(d_1)$, $y_1 \in [u_b, \sigma''(l_b); d_1) = 0$	$(l_b) = U \cdot l_b$	$\sigma'''(d_2)$, $l_b \in [y_1, \sigma''(u_b); d_2) = 0$	$\sigma''(u_b) = L \cdot u_b$
R_3	R_4	$\sigma'''(d_1)$, $y_3 \in [u_b, \sigma''(l_b); d_1) = 0$	$\sigma''(l_b) = U \cdot l_b$	$\sigma'''(d_2)$, $l_b \in [y_3, \sigma''(u_b); d_2) = 0$	$\sigma''(u_b) = L \cdot u_b$
R_2	R_3	$\sigma'''(d_1)$, $l_b \in [y_2, \sigma''(u_b); d_1) = 0$	$\sigma''(u_b) = U \cdot u_b$	$\sigma'''(d_2)$, $y_2 \in [u_b, \sigma''(l_b); d_2) = 0$	$\sigma''(l_b) = L \cdot l_b$
R_1	R_3	$\sigma'''(d_1) + (1-)\sigma'''(d_2)$, $l_b \in [y_{max}, \sigma''(l_b); d_1) = 0$, $y_{max} \in [u_b, \sigma''(u_b); d_2) = 0$	$U_1 + (1-)U_2$, $U_1 = (\sigma''(l_b) - \sigma'''(d_1))/(l_b - d_1)$, $U_2 = (\sigma''(u_b) - \sigma'''(d_2))/(u_b - d_2)$	$\sigma'''(d_3)$, $y_1 \in [u_b, \sigma''(l_b); d_3) = 0$	$\sigma''(l_b) = L \cdot l_b$
R_2	R_4	$\sigma'''(d_1)$, $l_b \in [y_2, \sigma''(u_b); d_1) = 0$	$\sigma''(u_b) = U \cdot u_b$	$\sigma'''(d_2) + (1-)\sigma'''(d_3)$, $l_b \in [y_{min}, \sigma''(l_b); d_2) = 0$, $y_{min} \in [u_b, \sigma''(u_b); d_3) = 0$	$L_1 + (1-)L_2$, $L_1 = (\sigma''(l_b) - \sigma'''(d_2))/(l_b - d_2)$, $L_2 = (\sigma''(u_b) - \sigma'''(d_3))/(u_b - d_3)$
R_1	R_4	$\sigma'''(d_1) + (1-)\sigma'''(d_2)$, $l_b \in [y_{max}, \sigma''(l_b); d_1) = 0$, $y_{max} \in [u_b, \sigma''(u_b); d_2) = 0$	$U_1 + (1-)U_2$, $U_1 = (\sigma''(l_b) - \sigma'''(d_1))/(l_b - d_1)$, $U_2 = (\sigma''(u_b) - \sigma'''(d_2))/(u_b - d_2)$	$\sigma'''(d_3) + (1-)\sigma'''(d_4)$, $l_b \in [y_{min}, \sigma''(l_b); d_3) = 0$, $y_{min} \in [u_b, \sigma''(u_b); d_4) = 0$	$L_1 + (1-)L_2$, $L_1 = (\sigma''(l_b) - \sigma'''(d_3))/(l_b - d_3)$, $L_2 = (\sigma''(u_b) - \sigma'''(d_4))/(u_b - d_4)$

this case we use the same bounding as in the case where $l_b \le 0 \le u_b$: h^U is the tangent line at $d_1 \le y_1$ that passes through $(l_b; \sigma'(l_b))$, and h^L is the tangent line at $d_2 \le y_1$ that passes through $(u_b; \sigma'(u_b))$. In a similar fashion, for the case in which $l_b \in R_2$ and $u_b \in R_3$, i.e., l_b in the concave region and u_b in the second convex region, we take the opposite approach: h^U is the tangent line at $d_1 \ge y_2$ that passes through $(u_b; \sigma'(u_b))$, and h^L is the tangent line at $d_2 \ge y_2$ that passes through $(l_b; \sigma'(l_b))$. These two cases are plotted in Figures 6a and 6b.

Finally, we tackle the case where $l_b \in R_1$ and $u_b \in R_3$, i.e., where l_b is in the first convex region and u_b is in the second convex region. Given there is a concave region in between them, two valid upper bounds would be the ones considered previously for $l_b \in R_1$ and $u_b \in R_2$, and $l_b \in R_2$ and $u_b \in R_3$. To obtain these bounds, we shift the upper bound in the first case to 0, and the lower bound in the second case to 0 (see h^U in Figure 6c). As our bounding requires a single h^U, we take a convex combination of the two bounds obtained. For the lower bound, we use a line that passes by either $(u_b; \sigma'(u_b))$, if $l_b \le u_b$, or by $(l_b; \sigma'(l_b))$, otherwise, as well as by a tangent point $d_3 \in R_1$, if $l_b \le u_b$, or by $d_4 \in R_3$, otherwise. See the line $h^{U;}$ in Figure 6c for a visual representation.

Bounding $\sigma''(y) = -2\tanh(y)(1 - \tanh(y)^2)$. By inspecting the derivative of σ'', $\sigma'''(y) = -2 + 8\tanh^2(y) - 6\tanh^4(y)$, we conclude there are three inflection points for this function, one at $y_1 = \arg\max_{y \le 0} \sigma'''(y)$, another at $y_2 = 0$, and finally at $y_3 = -y_1$. Take also, for the sake of bounding, $y_{max} = \arg\max_{y \ge 0} \sigma'''(y)$ and $y_{min} = \arg\min_{y \ge 0} \sigma'''(y)$.



(a) $l_b \in R_1$ and $u_b \in R_3$ (b) $l_b \in R_2$ and $u_b \in R_4$ (c) $l_b \in R_1$ and $u_b \in R_4$

Figure 7: Relaxing $\sigma''(y) = -2\tanh(y)(1-\tanh^2(y))$: examples of the linear relaxations of σ'' for different sets of l_b and u_b.

This leads to four different regions of σ'': $y \in]-\infty; y_1]$ (R_1, the first convex region), $y \in]y_1; y_2]$ (R_2, the first concave region), $y \in]y_2; y_3]$ (R_3, the second convex region), and $y \in]y_3; +\infty[$ (R_4, the second concave region). This leads to 10 combinations for the location of l_b and u_b.

The first four are straightforward: if $l_b \in R_i$ and $u_b \in R_i$ for $i \in \{1, \dots, 4\}$, then we use exactly the same approximations as for σ and σ'', varying only based on the convexity of R_i. Similarly, if $l_b \in R_i$ and $u_b \in R_{i+1}$ for $i \in \{1, 2, 3\}$, then we are also in the same situation as the adjacent regions of different convexity from σ, so we use exactly the same relaxation.

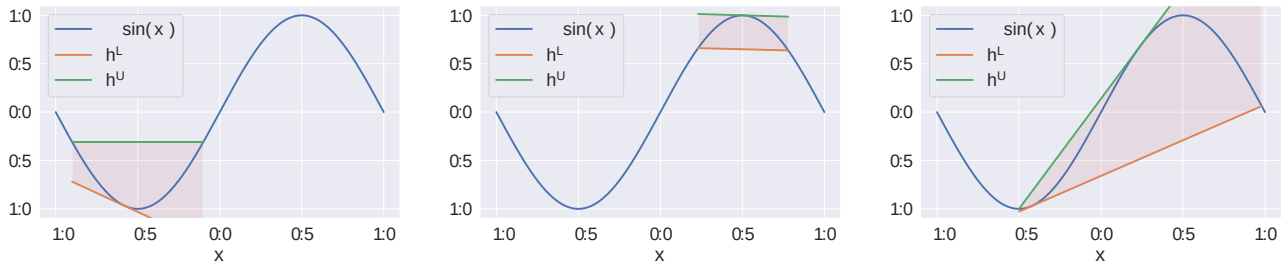
We are left with three cases where l_b and u_b are in non-adjacent regions. For $l_b \in R_1$ and $u_b \in R_3$, we are in the same scenario as in the bounding of σ, since R_1 and R_3 are convex regions separated by a concave one. In that case we follow the bounding procedure outlined before for σ - see Figure 7a for an example of it applied in this setting. For the case where $l_b \in R_2$ and $u_b \in R_4$, we are in an analogous case where R_2 and R_4 are concave regions separated by a convex one. As such, we consider the two valid lower bounds computed previously for $l_b \in R_2$ and $u_b \in R_3$, and $l_b \in R_3$ and $u_b \in R_4$. To obtain these bounds, we shift the upper bound in the first case so as to min $\sigma''(y)$, and the lower bound in the same case to the same value (see h^L in Figure 7b). As our bounding requires a single h^L, we take a convex combination of the two bounds, $h^{L,\cdot}$. For the upper bound, we simply assume l_b is in a concave region while u_b is in a convex region, and take the tangent at d for $\arg\max \sigma''(y)$ d 0 (see h^U in Figure 7b). Finally, we are left with the case where $l_b \in R_1$ and $u_b \in R_4$. In that case, we take the upper bound lines from the case where $l_b \in R_1$ and $u_b \in R_3$, and the lower bound ones from where $l_b \in R_2$ and $u_b \in R_4$. As before, given the requirement of one lower and upper bound functions, we take a convex combination of both in $h^{L,\cdot}$ and $h^{U,\cdot}$, respectively. See Figure 7c for a visual representation.

F. Linear lower and upper bounding nonlinear functions

Throughout, we assume the function's input x is lower bounded by l_b and upper bounded by u_b (i.e., $l_b \le x \le u_b$), and define the upper bound line as $h^U(x) = \lambda^U(x + \beta^U)$, and the lower bound line as $h^L(x) = \lambda^L(x + \beta^L)$. For the sake of brevity, we define for a function $h : \mathbb{R} \to \mathbb{R}$, and points $p, d \in \mathbb{R}$ the function $\Delta(h, p, d) = (h(p) - h(d))/(p - d) - h'(d)$. This is useful as for a given h and p, if there exists an $d \in [d_l, d_u]$, such that $\min_{d \in [d_l, d_u]} \Delta(h, p, d) = 0$, then $h'(d)$ is the slope of a tangent line to h that passes through p and d.

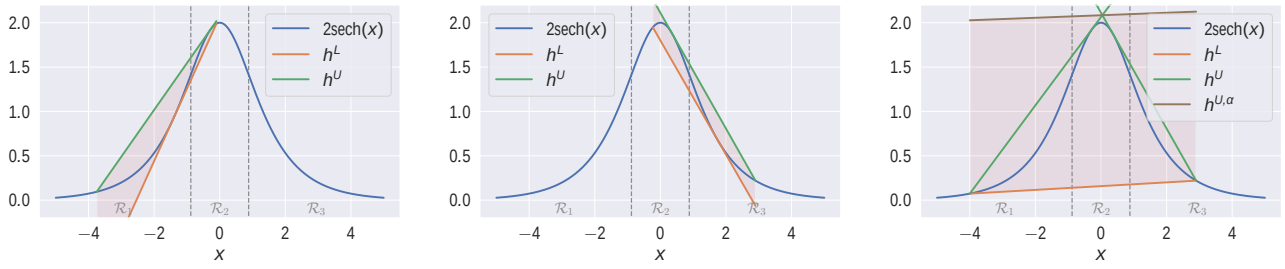
F.1. Case study: $\sin(x)$ for $x \in [-1, 1]$

As in Appendix E, we observe the convexity of the function $\sin(x)$ for $x \in [-1, 1]$, noticing that the function is convex for $x \le 0$ and concave for $x \ge 0$. For $l_b \le u_b \le 0$ we let h^U be the line that connects l_b and u_b, and for an arbitrary $d \in [l_b, u_b]$ we let h^L be the tangent line at that point. Similarly, for $0 \le l_b \le u_b$ we let h^L be the line that connects l_b and u_b, and for an arbitrary $d \in [l_b, u_b]$ we let h^U be the tangent line at that point. For the last case where $l_b \le 0 \le u_b$, we let h^U be the tangent line at $d_1 \le 0$ that passes through $(l_b, \sin(l_b))$, and h^L be the tangent line at $d_2 \ge 0$ that passes through $(u_b, \sin(u_b))$. Given the similarity of to the $\tanh$ bounds from Zhang et al. (2018), we omit a summary table, but present 3 examples of the possible cases in Figure 8.



(a) l_b 0 and u_b 0 (b) l_b 0 and u_b 0 (c) l_b 0 and u_b 0

Figure 8: *Relaxing* $-\sin(\pi x)$: examples of the linear relaxations for different sets of l_b and u_b.



(a) l_b 0 and u_b 0 (b) l_b 0 and u_b 0 (c) l_b 0 and u_b 0

Figure 9: *Relaxing* $2sech(x)$: examples of the linear relaxations for different sets of l_b and u_b.

F.2. Case study: $2\text{sech}(x)$ for $x \in [-5, 5]$

We start by observing that the function $2\text{sech}(x)$ is similar to the derivative of $\tanh$, whose relaxation we presented in Appendix E. By inspecting it's derivative, $f'(x) = 2\text{sech}(x)\tanh(x)$, we conclude that there are two inflection points at $x_1 = \max f'(x)$ and $x_2 = \min f'(x)$, leading to three different regions: $x \in]-\infty, x_1]$ ($\mathcal{R}_1$, the first convex region), $x \in]x_1, x_2]$ ($\mathcal{R}_2$, the concave region), and $x \in]x_2, +\infty[$ ($\mathcal{R}_3$, the second convex region). As a result, there are 6 combinations for the location of l_b and u_b which must be resolved. This is exactly the same case as the first derivative of $\tanh$, simply with x_1 and x_2 instead of y_1 and y_2. Due to the similarities, we can use exactly the same relaxations as presented in Table 3. We present visual examples of 3 cases of this relaxation in Figure 9.