Deterministic global optimization with trained neural networks: Is the envelope of single neurons worth it?

Clara Witte[1] and Jannik T. Lüthje[1] and Victor Schulte[1] and Alexander Mitsos[1] and Dominik Bongartz[2,*]

[1] Process Systems Engineering (AVT.SVT), RWTH Aachen University, 52074 Aachen, Germany
[2] Department of Chemical Engineering, KU Leuven, 3001 Leuven, Belgium

Abstract: Optimization problems containing trained neural networks remain challenging due to their nonconvexity. Deterministic global optimization relies on relaxations which should be tight, quickly convergent, and cheap to evaluate. While envelopes of common activation functions have been established for several years, the envelope of an entire neuron had not. Recently, Carrasco and Muñoz (arXiv.2410.23362, 2024) proposed a method to compute the envelope of a single neuron for S-shaped activation functions. However, the computational effectiveness of this envelope in global optimization algorithms is still unknown. Therefore, we implemented this envelope in our open-source deterministic global solver MAiNGO and machine-learning toolbox MeLOn, using the hyperbolic tangent activation function. We evaluate the benefit compared to combining the separate envelopes of the pre-activation and activation functions using factorable programming techniques in illustrative examples as well as case studies from chemical engineering. The results show that the use of the envelope slightly reduces the number of iterations but can strongly increase the computational time.

Keywords: *global optimization, artificial neural network, machine learning, convex envelope*

1 Introduction

Artificial neural networks (ANNs) are used in various applications such as (bio-)chemical engineering [8]. Therefore, integrating trained ANNs into optimization problems has become a point of interest [3, 11]. The presence of suboptimal local minima motivates the use of deterministic global optimization (DGO) algorithms. An essential aspect of DGO, which is usually based on branch-an-bound (B&B), is the construction of relaxations that are tight, quickly converging, and cheap-to-evaluate. Prime examples of popular methods applicable to a wide range of functions, including ANNs, are the (multivariate) McCormick technique [7, 15] or the auxiliary variable method (AVM) [7, 13]. Both are applicable to factorable functions, i.e., compositions of simpler, so-called *intrinsic functions*, including elementary operations. They rely on convex and concave relaxations of the intrinsic functions being available and provide a way to construct relaxations of the composition. Tighter relaxations for intrinsic functions result in tighter relaxations for the composition. Additionally, achieving a high convergence order of the relaxations is important to prevent unfavorable computational behavior, especially the cluster effect in B&B algorithms [5].

We [11] introduced the envelope of the hyperbolic tangent activation function and more importantly demonstrated that with McCormick relaxations, a reduced-space formulation of an ANN embedded in an optimization problem is advantageous compared to the full-space formulation where intermediates are treated as variables and additional equations as constraints, which are used to connect these intermediate variables. Wilhelm et al. [16] derived convex and concave envelopes for various other activation functions that are commonly used in ANNs.

However, even when using the envelope of the activation function, the relaxations obtained via the McCormick method or AVM for entire ANNs often remain weak. Even for a *single neuron*, which is the composition of the univariate activation function with an affine multivariate pre-activation function, these methods usually do not yield the convex envelope. To achieve tighter relaxations, Tjandraatmadja et al. [14] considered the multivariate input space of a neuron with the ReLU activation function to derive the envelope, thereby improving the performance of neural network verification. Recently, Carrasco and Muñoz [2] introduced a more general method for computing the convex and concave envelopes of a neuron with convex or S-shaped activation functions. They applied this approach in the context of AVM with a cutting-plane algorithm to improve the initial activation bounds of the neurons in the ANN.

However, to our knowledge, this new approach of computing the envelope of a neuron has not yet been applied in DGO in the open literature, and thus its effect on computational performance has not been investigated. To close this gap, we implemented the envelope proposed by Carrasco and Muñoz [2], within MC++ [4] for use in our open source optimizer McCormick-based Algorithm for mixed-integer Nonlinear Global Optimization (MAiNGO) [1], and our toolbox Machine Learning models for Optimization (MeLOn) [12]. Note that we use McCormick method, which operates on the original

Corresponding author: *Dominik Bongartz
Department of Chemical Engineering, KU Leuven, Belgium
E-mail: dominikbongartz@alum.mit.edu

variables. We then compare the application of the envelope of a neuron to the application of the separate envelopes of pre-activation and activation functions, focusing on the tightness of relaxations, as well as computational time and number of iterations for solving optimization problems involving embedded ANNs globally.

2 Numerical results

We consider multilayer perceptrons (MLPs), and select the hyperbolic tangent as activation function. The approach proposed by Carrasco and Muñoz [2] will be referred to as the *envelope of a neuron*. In contrast, the approach utilizing the McCormick relaxations will be referred to as the *envelope of activation function*; more precisely, for a neuron, we will use the envelope of the activation function [11], the relaxation of the affine pre-activation function, and the univariate composition theorem of McCormick [7]. For both approaches, we use the multivariate McCormick composition theorem [15] for the composition of neurons in an MLP.

We used our open-source deterministic-global optimization solver MAiNGO v0.8.3 [1]. It is based on a B&B, implemented in C++, and uses the open-source library MC++ [4] to compute convex and concave relaxations. For this work, we extended our fork of MC++[1] to include the envelope of a neuron with the hyperbolic tangent as activation function. We used CPLEX v12.9 to solve the lower bounding problem. For our experiments, we used various bound tightening techniques corresponding to MAiNGO's default settings, except for constraint propagation, as its omission resulted in faster computation for our case studies. Furthermore, we investigated the effect of using subgradients to tighten interval bounds on intermediate factors as proposed by Najman and Mitsos [10]. The MATLAB Deep Learning Toolbox was used via a script provided by MeLOn [12] to train neural networks and subsequently integrate these models into MAiNGO. All case studies were conducted on an Intel Core i5-10500 with 3.1GHz, 32 GB RAM, running Windows 10. The case studies, including all MLP parameters, are provided online[2].

2.1 Illustrative example and scaling with network size

As an illustrative example, we approximated the two-dimensional peaks function available in MATLAB using an MLP and subsequently minimized it, as also done in [11]. The peaks function is defined as $f_{\text{peaks}} : \mathbb{R}^2 \to \mathbb{R}$ with

$$\begin{aligned} f_{\text{peaks}}(x_1, x_2) = {} & 3\,(1 - x_1)^2\, \mathrm{e}^{-x_1^2 - (x_2+1)^2} \\ & - 10 \left(\frac{x_1}{5} - x_1^3 - x_2^5 \right) \mathrm{e}^{-x_1^2 - x_2^2} - \frac{\mathrm{e}^{-(x_1+1)^2 - x_2^2}}{3}. \end{aligned}$$

The function exhibits multiple local minima within the domain $D = [-2, 2] \times [-2, 2] \in \mathbb{R}^2$ but only one global minimum. Given sufficiently accurate training, the MLP shares these properties. To learn the peaks function, 1000 data points within D were generated using Latin hypercube sampling. These data points were randomly split into training, validation, and test sets with a $0.7 : 0.15 : 0.15$ ratio, and the MLP weights were fitted in MATLAB's Deep Learning Toolbox by minimizing the mean-squared error on the training set and the early stopping procedure. As activation function, the hyperbolic tangent was used for each hidden layer. A linear activation function was used for the output layer. Multiple MLP architectures were trained while varying numbers of hidden layers as well as the number of neurons per hidden layer. Each configuration was trained 10 times. Once trained, the MLP was integrated into an optimization problem to minimize the output of the MLP as a function of its inputs using the aforementioned reduced-space formulation [11]. The optimizations were solved with an absolute and relative optimality tolerance of 10^{-4} and a maximum runtime of 3600 s. Each MLP was optimized independently five times, and the mean and standard deviation of the CPU times were calculated for each MLP architecture.

First, we examine the tightness of the convex and concave relaxations of the MLPs, comparing the relaxations that use the envelope of a neuron $f^{\text{cv/cc,neuron}}$ with those that use the envelope of the activation function $f^{\text{cv/cc,tanh}}$, initially without using the tightening via subgradients available in MAiNGO. For a network with one hidden layer, the relaxations that use the envelope of a neuron are tighter than the relaxations that use the envelope of the activation function (blue vs. red lines in Fig. 1a–1b). At the center of the domain ($x_1 = x_2 = 0$), both relaxations yield identical values (see Fig. 1a). However, the gap between the relaxations becomes larger towards the edges, with the most significant differences

[1] https://git.rwth-aachen.de/avt-svt/public/thirdparty/mcpp. Accessed 12 February 2025
[2] http://permalink.avt.rwth-aachen.de/?id=631600

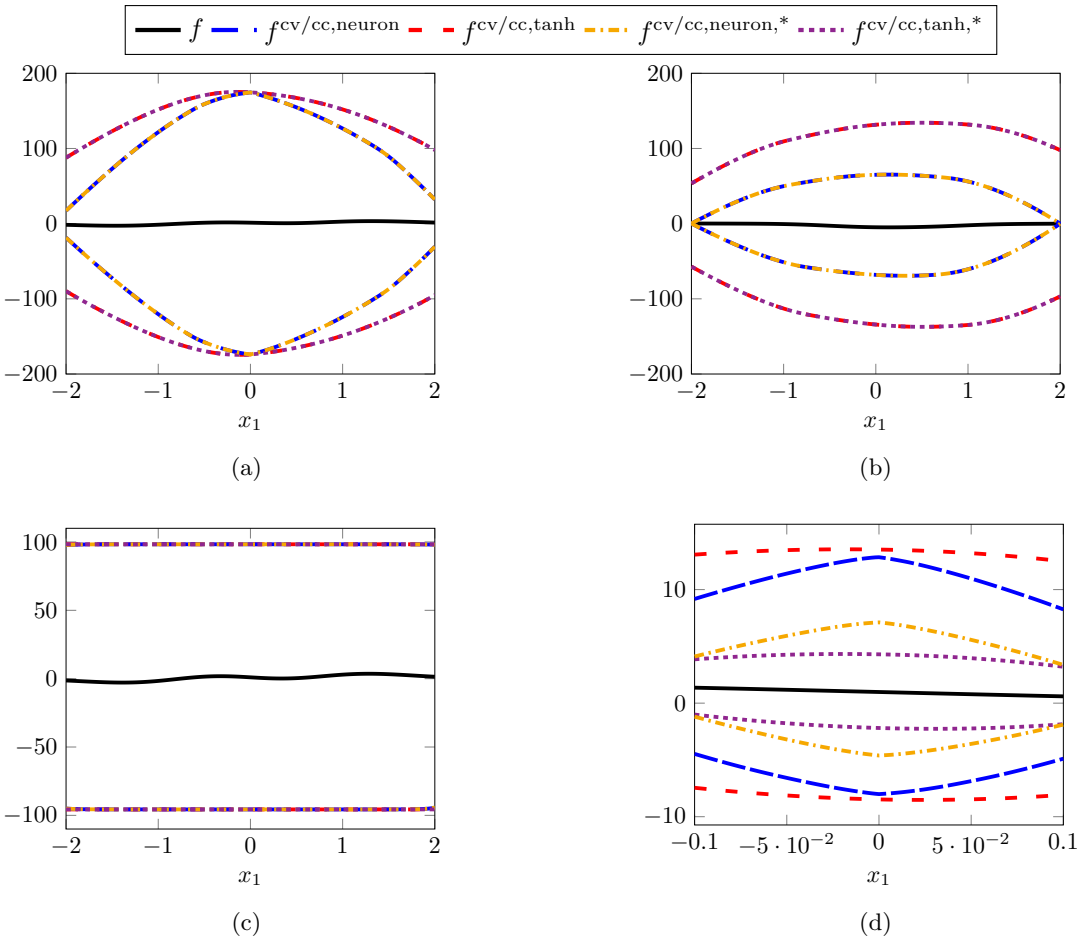

Figure 1: Illustration of the tightness of convex and concave relaxations of an MLP. The relaxations for one hidden layer ((a)-(b)) are evaluated over the domain $D = [-2, 2]^2 \in \mathbb{R}^2$ and plotted over the interval $x_1 \in [-2, 2]$ while fixing x_2. Cross-sections are shown for (a) $x_2 = 0$ and (b) $x_2 = -2$. For the one hidden layer MLP, the relaxations of the MLP output f constructed using the envelope of a neuron $f^{\text{cv/cc, neuron}}$ are tighter than the relaxations constructed using the envelope of the activation function $f^{\text{cv/cc,tanh}}$, particular near the corners. For the three hidden layers MLP ((c)-(d)), cross-sections are shown for $x_2 = 0$, with relaxations evaluated over the domain (c) D and (d) $\tilde{D} = [-0.1, 0.1]^2 \in \mathbb{R}^2$. The relaxations using the envelope of the activation function with tightening based on subgradients $f^{\text{cv/cc,tanh,*}}$ can be even tighter than those using the envelope of a neuron with the same tightening $f^{\text{cv/cc,neuron,*}}$

occurring at the corner points (see Fig. 1b). This is in line with the finding of Carrasco and Muñoz [2], who observed similar improvements in the tightness of relaxations. However, as the number of hidden layers in the MLP increases, the gap between the two relaxations tends to diminish, as shown in Fig. 1c for an MLP with three hidden layers: there is no observable difference between the two relaxations, which appear almost flat. A gap occurs only near the boundaries, where the relative difference between the relaxations is less than 1 %. This phenomenon can be attributed to the characteristics of the hyperbolic tangent activation function: The hyperbolic tangent becomes saturated for large input values. As the input bounds of deeper hidden layers in MLPs get wider, the relaxations become flatter and less tight overall. When evaluated on a smaller domain, the differences between the relaxations become more pronounced again (blue vs. red lines in Fig. 1d), which is relevant as the node domain shrinks due to branching and bound tightening in B&B algorithms.

Next, we investigate the impact of tightening interval bounds on intermediate factors based on subgradients [10], which is available in MAiNGO. The impact of this tightening depends on the MLP architecture: For a single hidden layer, the tightening has no effect (see Fig. 1a–1b). In this case, the relaxations of neurons are only constructed over a box domain, the inputs of these neurons are independent, and the natural interval extensions are hence exact, such that no tightening is possible. However, from the second hidden layer onward, the relaxations are constructed over a set defined by the relaxations of the previous

hidden layers, allowing the tightening to have an impact. Indeed, for the present example, the tightening based on subgradients allows the relaxations to become tighter than those without tightening (orange and purple lines in Fig. 1d). Moreover, when using this tightening, we observe that the relaxations using the envelope of the activation function are *tighter* than the relaxations using the envelope of a neuron (see Fig. 1d). This is due to the fact that tighter interval bounds can be achieved via the tightening for the sum term in the affine pre-activation function, which then results in tighter relaxations using the envelope of the activation function. In contrast, for the envelope of a single neuron, no interval bounds of the pre-activation function are considered when computing the relaxations. Here, the tightening based on subgradients cannot tighten the relaxation of the neuron itself but only tighten its interval bounds, which then improves the tightness of the relaxations only in the subsequent layers.

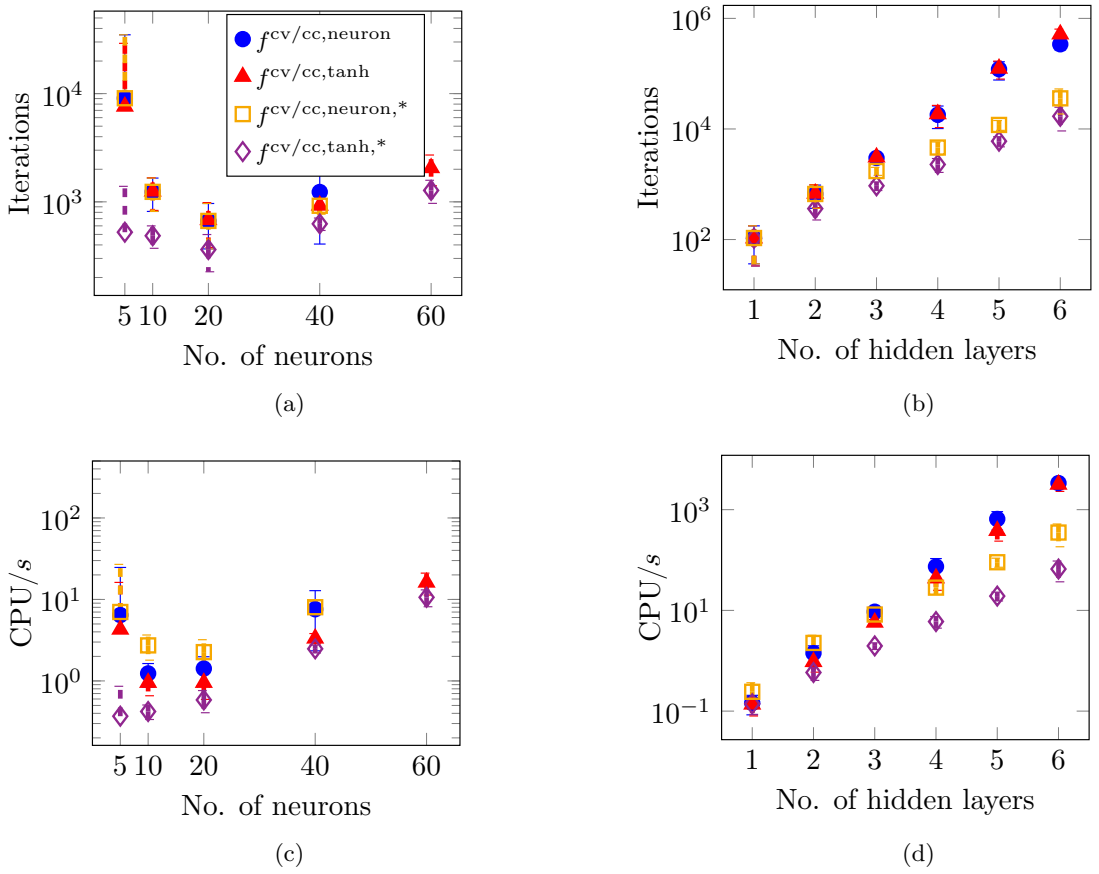

Figure 2: CPU times and number of iterations for minimizing MLPs trained on the peaks function using the midpoint linearization method for different MLP sizes. For (a), (c) we vary the number of neurons per layer with two fixed hidden layers; for (b), (d) we instead vary the number of hidden layers with a fixed number of neurons (20) in each hidden layer. Using the envelope of a neuron $f^{cv/cc,\,neuron}$ can reduce the number of iterations but increases CPU time compared to the envelope of activation function $f^{cv/cc,tanh}$. Relaxations using the envelope of the activation function with tightening based on subgradients $f^{cv/cc,tanh,*}$ outperform those using the envelope of a neuron with the same tightening $f^{cv/cc,neuron,*}$. The configuration with two hidden layers and 60 neurons per hidden layer reached the CPU time limit when using the envelope of a neuron; therefore, this case is not shown in the plots

After tightness of relaxations, the peaks function is used to illustrate the scaling of CPU time and B&B iterations for solving the optimization problems with the MLP size. We investigate the effect of two MLP hyperparameters: (a) the number of hidden layers (between 1 and 6), with a fixed number of neurons (20) per hidden layer, and (b) the number of neurons (between 5 and 60) per hidden layer, with a fixed number of hidden layers (2). Without tightening based on subgradients, the use of the envelope of neurons requires slightly fewer B&B iterations than the envelope of the activation function (see Fig. 2b). In this sense, the optimization can thus benefit from the tighter relaxations. However, this trend of fewer B&B iterations using the envelope of a neuron is not observed when the number of neurons per hidden layer increases (see Fig. 2a). Furthermore, the optimization using the envelope of a neuron requires *more*

CPU time than when using the envelope of the activation function, regardless of the number of hidden layers and neurons (see Fig. 2c–2d). This is due to the fact that the computation of the envelope of a neuron is more expensive, as it requires solving n one-dimensional envelope computations in the worst case, where n is the number of inputs to the neuron [2].

Since MAiNGO (similar to most DGO solvers) does not use the convex relaxations themselves for bounding but rather constructs linear programs by linearization, we also investigated the effect of another linearization strategy, i.e., we compared the default *midpoint* strategies to the *Simplex-Kelley* strategy [9]. The midpoint strategy uses a single linearization point at the center of the B&B node. In contrast, the Simplex-Kelley strategy starts with $n + 1$ linearization points arranged in a simplex for a problem with n variables and iteratively adds points following the Kelley algorithm. The trends observed with Simplex-Kelley remained the same as those with the midpoint strategy shown above. Simplex-Kelley did achieve a larger relative reduction in iterations, and using the envelope for a neuron required fewer iterations for MLPs with 5 and 40 neurons per hidden layer. Still, the CPU time remained larger than when using the envelope of the activation function.

Finally, when using the tightening based on subgradients, we observe a significant reduction in the CPU time and B&B iterations for the optimization, with a reduction up to two orders of magnitude for the deepest MLP (see Fig. 2c–2b). As previously shown, using this tightening yields tighter relaxations, which can explain the observed improvements in optimization performance. Thus, in the present implementation, the fastest option is still to use the envelope of the activation function together with the subgradient tightening.

2.2 Engineering applications

To evaluate the effect of the relaxations in more practical problems, we consider three optimization problems with embedded MLPs from engineering applications. The problems differ from the peaks function in terms of input and output size, and in that some of them involve multiple MLPs (see Table 1). From our previous work [11], we consider the optimization of glucose fermentation to gluconic acid based on experimental data and the optimization of the operating point for a compressor plant. Our final case study is a reduced-space model of a Carnot battery based on our recent work [6], where we maximize the round-trip efficiency of this energy storage process using MLPs to calculate thermodynamic properties. For all optimizations, we applied tightening based on subgradients. For the fermentation process and compressor plant, we used the settings described above. For the Carnot battery, we used default MAiNGO settings, which included constraint propagation as it was computationally beneficial in this case, and an optimality tolerance of 10^{-2} to allow convergence within the time limit. Again, we ran the case studies five times and calculated the mean CPU time.

For the fermentation case study, the envelope of a neuron performed similarly to the envelope of the activation function in terms of CPU time (see Fig. 3a) and B&B iterations (see Fig. 3b). For the compressor case study, the envelope of a neuron performed worse than the envelope of the activation function regarding CPU time and B&B iterations, with a relative difference of approximately 39 % in CPU time and 26 % in B&B iterations. In this case study, tightening based on subgradients can impact the MLPs with three hidden layers, reducing iterations and CPU time. In contrast, for the Carnot battery, the envelope of a neuron needed 2.55 % less B&B iterations compared to using the envelope of the activation function. However, it required 2.48 % more CPU time compared to using the envelope of the activation function. In conclusion, while the envelope of a neuron can be advantageous, its computational performance differs from case to case.

Table 1: MLP architecture of the case studies of [6, 11], where $k^{(i)}$ denotes the number of neurons in layer i, and n the total number of layers

Case study	Number of MLPs	Number of hidden layers	$k^{(1)}$	$k^{(i)}$ with $i = 2, ...n-1$	$k^{(n)}$
Fermentation process [11]	1	2	3	2	1
Compressor plant [11]	2	3	2	10	1
Carnot battery [6]	18	1	1-2	5-9	1

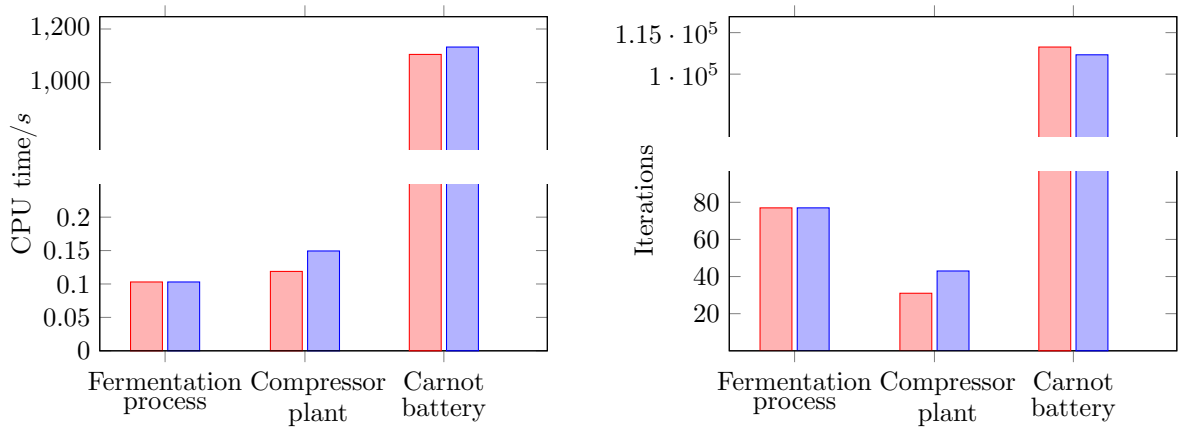

Figure 3: CPU times and number of iterations for solving three engineering case studies using the envelopes of activation function (red) versus the envelopes of a neuron (blue). Which method works best depends on the case study

3 Conclusion and outlook

We implemented the envelope proposed by Carrasco and Muñoz [2] for neurons with hyperbolic tangent activation functions within our open-source deterministic global solver MAiNGO, which utilizes the (multivariate) McCormick method to construct relaxations. Our results confirm that the envelope of a neuron offers tighter relaxations of MLPs than merely using the envelope of the activation function. Additionally, we show that this approach is particularly beneficial in terms of reducing B&B iterations in the context of deeper neural networks. However, it often does not reduce, and in fact may increase, the CPU time in the current implementation. This is due to the higher cost of computing the envelope of a neuron. Furthermore, we demonstrate significant speedups with our subgradient-based tightening from [10]. This tightening yields tighter relaxations and lower CPU time compared to the relaxations without it. Moreover, we show that this tightening has a greater impact when using the envelope of the activation function, as the tighter interval bounds of the pre-activation function can be translated into tighter relaxations of the neuron output. In contrast, when using the envelope of a neuron, the tightening only affects the interval bounds on the neuron output and can thus only provide tighter relaxations in subsequent layers.

Our main conclusion is that developing further relaxations of single neurons remains an open challenge: in addition to being tight, they must converge quickly and be computationally cheap. It is also conceivable to develop custom relaxations over multiple layers or even entire MLPs.

Author contributions

CW conceptualized the research, ran the simulations and wrote the manuscript. CW and DB interpreted the results, while all co-authors edited the manuscript. CW and VS implemented the envelope into the software. JTL aided with the adaption of the envelope to the software and provided one engineering application. DB and AM are the principal investigators and guided the effort.

Bibliography

[1] D. Bongartz, J. Najman, S. Sass, and A. Mitsos. *MAiNGO: McCormick based Algorithm for mixed integer Nonlinear Global Optimization*. Tech. rep. http://permalink.avt.rwth-aachen.de/?id=729717. Process Systems Engineering (AVT.SVT), RWTH Aachen University, 2018.

[2] P. Carrasco and G. Muñoz. "Tightening convex relaxations of trained neural networks: a unified approach for convex and S-shaped activations". In: *arXiv* (2024). https://doi.org/10.48550/arXiv.2410.23362.

[3] F. Ceccon, J. Jalving, J. Haddad, A. Thebelt, C. Tsay, C. D. Laird, and R. Misener. "OMLT: Optimization & Machine Learning Toolkit". In: *J. Mach. Learn. Res.* 23.349 (2022), pp. 1–8.

[4] B. Chachuat, B. Houska, R. Paulen, N. Peri'c, J. Rajyaguru, and M. E. Villanueva. "Set-Theoretic Approaches in Analysis, Estimation and Control of Nonlinear Systems". In: *IFAC-PapersOnLine* 48.8 (2015). https://github.com/omega-icl/mcpp, pp. 981–995.

[5] R. B. Kearfott and K. S. Du. "The Cluster Problem in Global Optimization: The Univariate case". In: *Computing (Suppl.)* 9 (1992), pp. 117–127.

[6] J. T. Lüthje, M. Langiu, and A. Mitsos. "Working Fluid Screening for ORC-Based Carnot Batteries by Deterministic Global Optimization of Design and Nominal Operation". In: (2024). Submitted for publication.

[7] G. P. McCormick. "Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems". In: *Math. Program.* 10.1 (1976), pp. 147–175.

[8] M. Mowbray, T. Savage, C. Wu, Z. Song, B. A. Cho, E. A. Del Rio-Chanona, and D. Zhang. "Machine learning for biochemical engineering: A review". In: *Biochem. Eng. J.* 172 (2021), p. 108054.

[9] J. Najman, D. Bongartz, and A. Mitsos. "Linearization of McCormick relaxations and hybridization with the auxiliary variable method". In: *J. Global Optim.* 80.4 (2021), pp. 731–756.

[10] J. Najman and A. Mitsos. "Tighter McCormick relaxations through subgradient propagation". In: *J. Global Optim.* 75.3 (2019), pp. 565–593.

[11] A. M. Schweidtmann and A. Mitsos. "Deterministic Global Optimization with Artificial Neural Networks Embedded". In: *J. Optimiz. Theory App.* 180.3 (2019), pp. 925–948.

[12] A. M. Schweidtmann, L. Netze, and A. Mitsos. *MeLOn - Machine Learning Models for Optimization*. https://git.rwth-aachen.de/avt-svt/public/MeLOn. Accessed: 12.02.2025. 2020.

[13] E. M. Smith and C. C. Pantelides. "Global optimisation of nonconvex MINLPs". In: *Comput. Chem. Eng.* 21 (1997), S791–S796.

[14] C. Tjandraatmadja, R. Anderson, J. Huchette, W. Ma, K. Patel, and J. P. Vielma. "The Convex Relaxation Barrier, Revisited: Tightened Single-Neuron Relaxations for Neural Network Verification". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 21675–21686.

[15] A. Tsoukalas and A. Mitsos. "Multivariate McCormick relaxations". In: *J. Global Optim.* 59.2-3 (2014), pp. 633–662.

[16] M. E. Wilhelm, C. Wang, and M. D. Stuber. "Convex and concave envelopes of artificial neural network activation functions for deterministic global optimization". In: *J. Global Optim.* 85.3 (2023), pp. 569–594.