Deterministic global optimization with trained neural networks: What is the benefit of the envelope of single neurons?

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

[1]Process System Engineering, RWTH Aachen University, Germany.
[2*]Department of Chemical Engineering, KU Leuven, Belgium.

*Corresponding author(s). E-mail(s): dominikbongartz@alum.mit.edu;

Abstract

Optimization problems containing trained neural networks remain challenging due to their nonconvexity. Deterministic global optimization relies on relaxations that 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 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 and Scaled Exponential Linear Unit as activation functions. 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 suggest that using the envelope of a neuron can provide tighter relaxations and reduce both the number of iterations as well as the computational time. However, if performing bound tightening based on subgradients, the separate envelopes often perform better as they allow tightening bounds on the preactivation output.

Keywords: Global optimization, Artificial neural network, Machine learning, Convex Envelope

1 Introduction

Artificial neural networks (ANNs) are used as data-driven models in various applications such as (bio-)chemical engineering [1]. Therefore, integrating trained ANNs into optimization problems has become a point of interest [2, 3]. 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-and-bound (B&B), is the construction of relaxations that are tight, quickly converging, and cheap-to-evaluate. Examples of popular methods applicable to a wide range of functions, including ANNs, are the (multivariate) McCormick technique [4, 5] or the auxiliary variable method (AVM) [4, 6]. 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 [7].

To provide a clearer understanding of how to construct a relaxation of an ANN embedded within an optimization problem, we first examine how a neural network – specifically a multilayer perceptron (MLP) – can be formulated mathematically when integrated into an optimization problem. MLPs are a special case of a feed-forward neural network and, as described in [8], MLPs can be described as a directed acyclic graph connecting N layers, numbered from $k = 1$ to N. It contains three types of layers: the input layer ($k = 1$), the output layer ($k = N$), and the hidden layers ($k = 2, ..., N-1$). In each layer, there are $D^{(k)}$ neurons. The output of a single neuron i in layer k ($k \geq 2$) can be described by the *single-neuron function* $z_i : \mathbb{R}^n \to \mathbb{R}$, defined as

$$g_i(\boldsymbol{x}) := \boldsymbol{w}_i\boldsymbol{x} + b_i \ , \tag{1}$$

$$z_i(\boldsymbol{x}) := \sigma_i(g_i(\boldsymbol{x})) \ . \tag{2}$$

The function $g_i : \mathbb{R}^n \to \mathbb{R}$ is called the pre-activation function. Here, the input vector $\boldsymbol{x}$ contains the output of the neurons in layer $k-1$. The vector $\boldsymbol{w}_i$ holds the constant weights, and b_i is a constant bias of the neuron i of the layer k; both are learned during training, but they are constant when the trained MLP is embedded in an optimization problem, which we consider herein. The function $\sigma_i : \mathbb{R} \to \mathbb{R}$ represents the activation function of the neuron i. In this work, we use the same activation function for all hidden layers and a linear function for the output layer. Thus, the MLPs we consider are compositions of several single neuron functions of the form z_i that differ in the values of the weights and biases.

One way to compute the convex relaxation of the single neuron function z_i is to treat it as a composition of the affine pre-activation function with the univariate activation function. We [2] introduced the envelope of the hyperbolic tangent activation function. More importantly, we demonstrated that with McCormick relaxations, a reduced-space formulation of an MLP embedded in an optimization problem is advantageous compared to the full-space formulation. In the reduced-space formulation, the layers of the MLP are evaluated sequentially to compute their relaxations in the space of the input variables. In the full-space formulation, in contrast, the pre-activation and activation outputs from each neuron are treated as intermediate variables, and the additional Equations (1) and (2) are used as constraints to connect these intermediate variables. Wilhelm et al. [9] then derived 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 MLPs 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. [10] 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 [11] 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 MLP.

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 [11], within MC++ [12] for use in our open source optimizer McCormick-based Algorithm for mixed-integer Nonlinear Global Optimization (MAiNGO) [13], and our toolbox Machine Learning models for Optimization (MeLOn) [14]. Note that we use the McCormick method, which operates on the original variables without introducing auxiliary 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 MLPs to global optimality.

2 Numerical Results

As described above, we focus on MLPs and select either the hyperbolic tangent or Scaled Exponential Linear Unit (SELU) as the activation function for all hidden layers of an MLP. The approach proposed by Carrasco and Muñoz [11] will be referred to as the *envelope of a neuron*; in this case, the single-neuron function z_i is thus considered as an intrinsic function in the factorable representation of the MLP. In contrast, the approach utilizing McCormick relaxations will be referred to as the *envelope of activation function*; more precisely, in this latter case, we use the envelope of the activation function σ_i [2], the relaxation of the affine pre-activation function g_i, and the univariate composition theorem of McCormick [4] to relax each single-neuron function z_i. For both approaches, we use the multivariate McCormick composition theorem [5] for the composition of neurons in an MLP.

We used our open-source deterministic-global optimization solver MAiNGO v0.10.1 [13]. It is based on a B&B algorithm, implemented in C++, and uses the open-source library MC++ [12] 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 v22.1 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 [15]. TensorFlow [16] was used via a script provided by MeLOn [14] to train neural networks and subsequently integrate these models into MAiNGO. All case studies were conducted on an Intel(R) Xeon(R) Gold 6544Y with 3.6GHz, 512 GB RAM, running Windows 11. All calculations were performed on a single core. The case studies, including all MLP parameters, are provided online[2].

2.1 Illustrative example and scaling with network size

2.1.1 Method

As an illustrative example, we approximated the two-dimensional peaks function using an MLP and subsequently minimized it, as also done in [2]. 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 TensorFlow by minimizing the mean-squared error on the training set and the early stopping procedure. Multiple MLP architectures were trained while varying the number of hidden layers as well as the number of neurons per hidden layer. We used the hyperbolic tangent as activation function for each hidden layer, and subsequently repeated the same set of computations using SELU as activation function. A linear activation function was used for the output layer. Each configuration was trained 10 times with different random seeds for weight and bias initialization. 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 [2]. The optimizations were solved over the bounded domain D with an absolute and relative optimality tolerance of 10^{-4}. Each MLP was optimized independently five times, and the mean and standard deviation of the CPU times were calculated for each MLP architecture. In this short communication, we present only the results obtained

[1] https://git.rwth-aachen.de/avt-svt/public/thirdparty/mcpp. Accessed March 2026
[2] http://permalink.avt.rwth-aachen.de/?id=631600

using the hyperbolic tangent as an activation function. The corresponding results for the SELU activation function are provided in the supplementary information (SI), and only qualitative similarities and differences are highlighted in the following.

2.1.2 Tightness of relaxations

First, we examine the tightness of the convex and concave relaxations of the MLPs, comparing the relaxations that use the envelope of a neuron, denoted as $f^{\mathrm{cv/cc,neuron}}$, with those that use the envelope of the activation function, denoted as $f^{\mathrm{cv/cc,tanh}}$, 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 occurring at the corner points (see Fig. 1b). This is in line with the finding of Carrasco and Muñoz [11], 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.

In general, a similar behavior is observed when using SELU as activation function (see Fig. S1 in SI). However, for the MLP with three hidden layers and wider input bounds, the resulting relaxations are less flat and have a larger gap between the relaxations of the different relaxation strategies (see Fig. S1c in SI). This difference can be attributed to the properties of SELU, which is unbounded and therefore not saturated for large input values.

Next, we investigate the impact of the tightening method based on subgradients [15] available in MAiNGO. In this method, the subgradients of the McCormick relaxations are used at each factorable representation of the problem to (potentially) tighten the interval bounds of that factor. When using the envelope of a neuron, this tightening is thus conducted for the output of each neuron. When using the envelope of activation function, the tightening can additionally be applied to the output of the pre-activation function, which is a separate factor in this case. 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 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

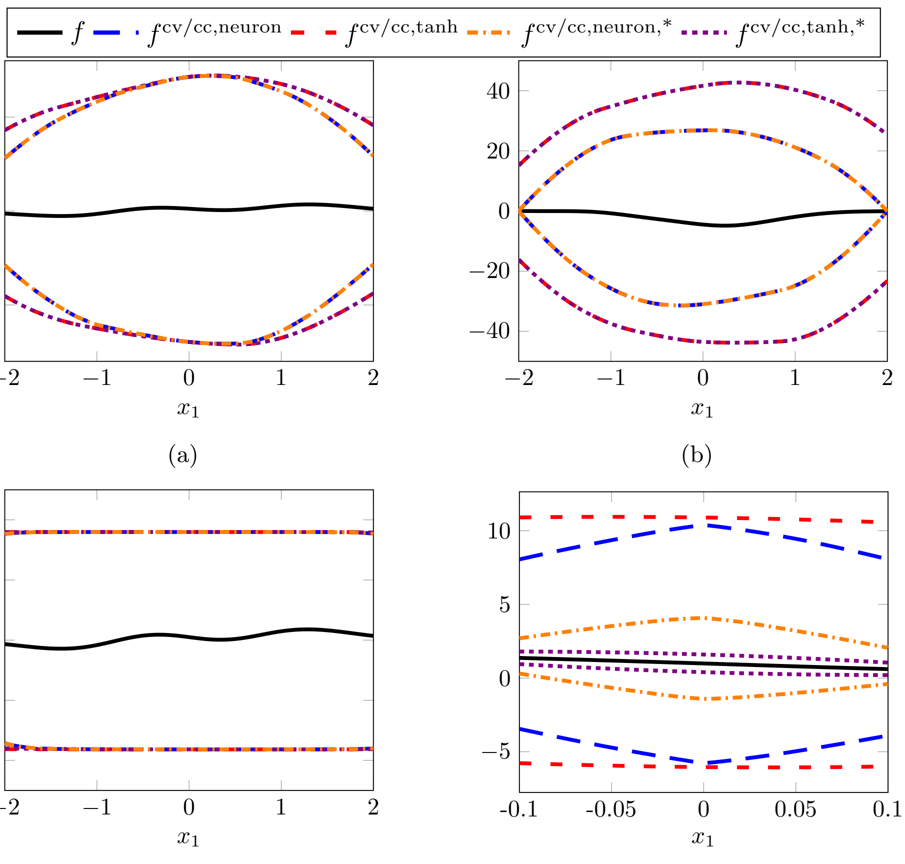


Fig. 1: Illustration of the tightness of convex and concave relaxations of an MLP with the hyperbolic tangent as activation function. The relaxations for an MLP with one hidden layer ((a)-(b)) are evaluated over the domain $D = [-2, 2]^2 \in \mathbb{R}^2$ and plotted over the interval $x_1 \in [-2, 2]$ 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 $f^{\text{cv/cc,neuron,*}}$

relaxations of the previous hidden layers (which, in general, is not a box), 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, since these are included in the single-neuron function, which is relaxed as a whole. 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.

2.1.3 Computational results

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 (blue circles vs. red triangles in Fig. 2b). For example, for the largest MLP with 6 hidden layers, the use of the envelope of a neuron reduces the number of B&B iterations the most by 3.4 % on average. In this sense, the optimization can thus benefit from the tighter relaxations. However, the trend of fewer B&B iterations when using the envelope of a neuron is not observed as the number of neurons per hidden layer increases. Instead, the iterations of both envelopes require almost the same number of iterations, differing by 1 % or less on average (blue circles vs. red triangles in Fig. 2a). In terms of computational time, the optimization using the envelope of a neuron generally requires equal or less CPU time than when using the envelope of the activation function, regardless of the number of hidden layers and neurons (blue circles vs. red triangles in Fig. 2c–2d). For example, the MLP with 6 hidden layers, using the envelope of a neuron reduces the runtime by 9.7% on average compared to the envelope of activation function. For a similarly large MLP with 2 hidden layers and 60 neurons per layer, the envelope of neuron is on average even 34 % faster. A similar trend is observed for the SELU activation function, where, in a MLP with 6 hidden layers, the use of the envelope of a neuron reduces the CPU runtime by 70 % on average compared to the envelope of the activation function (see Fig. S2 in SI).

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 [17]. 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

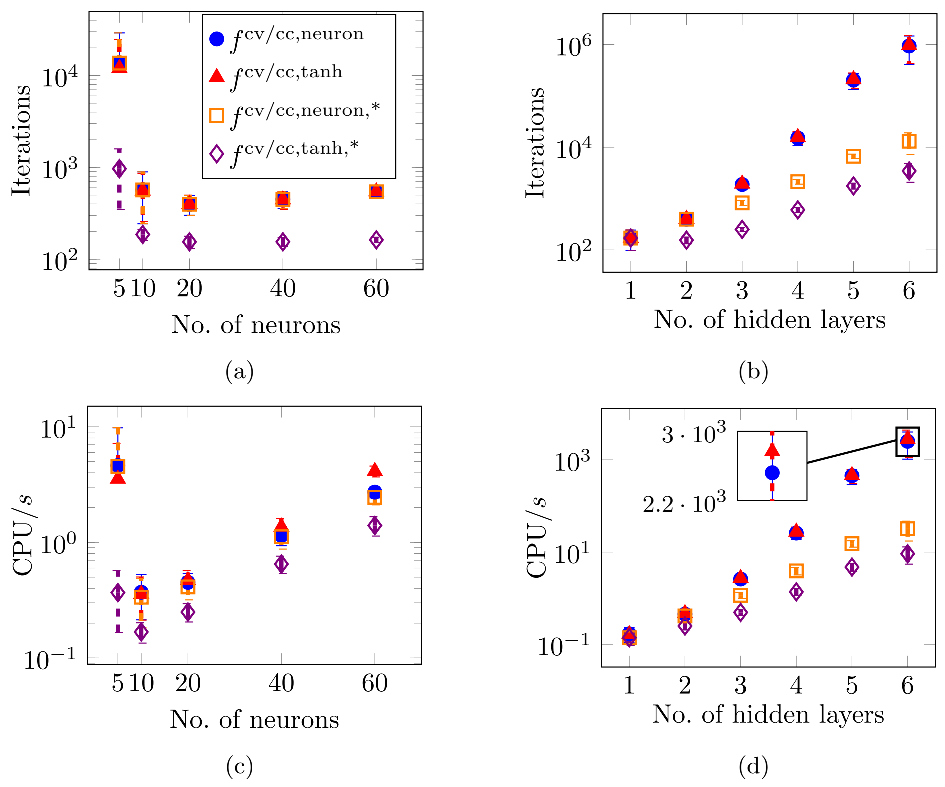


Fig. 2: CPU times and number of iterations for minimizing MLPs trained on the peaks function using the midpoint linearization method for different MLP sizes and hyperbolic tangent as activation function. For (a), (c), we vary the number of neurons per layer with two fixed hidden layers; for (b), (d), 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 $f^{\mathrm{cv/cc,\ neuron}}$ can reduce slightly the number of iterations and the CPU time compared to the envelope of activation function $f^{\mathrm{cv/cc,tanh}}$. Relaxations using the envelope of the activation function with tightening based on subgradients $f^{\mathrm{cv/cc,tanh,*}}$ outperform those using the envelope of a neuron with the same tightening $f^{\mathrm{cv/cc,neuron,*}}$.

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 achieved a larger relative reduction in iterations and CPU time for the envelope of a neuron compared to the envelope of an activation function (see Fig. S3 in the SI).

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. 2a–2d). This reduction is more pronounced for the envelope of the activation function, resulting in fewer B&B

Table 1: MLP architecture of the case studies of [2, 18], 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 [2]	1	2	3	2	1
Compressor plant [2]	2	3	2	10	1
Carnot battery [18]	18	1	1-2	5-9	1

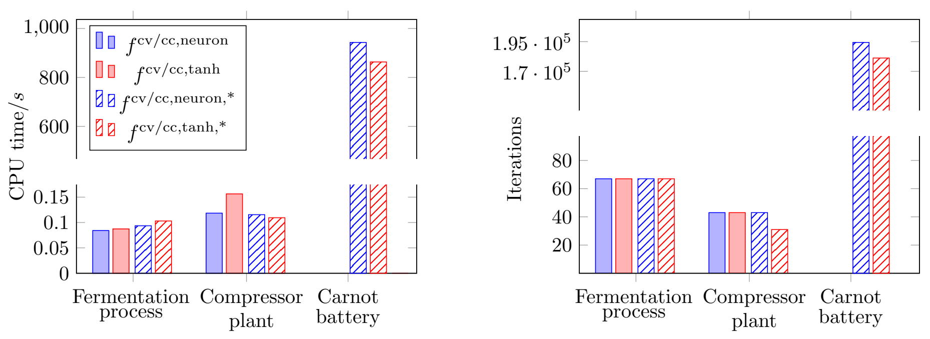
iterations and less CPU time compared to the envelope of a neuron (orange square vs. purple diamond in Fig. 2a–2d). As previously shown, using this tightening yields tighter relaxations, especially when using the separate envelopes of pre-activation and activation function, thanks to the possibility to tighten bounds on the pre-activation. This 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). All MLPs applied in the case studies use the hyperbolic tangent as activation function for each hidden layer.

From our previous work [2], 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 [18], where we maximize the round-trip efficiency of this energy storage process using MLPs to calculate thermodynamic properties. 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 we set an optimality tolerance of 10^{-2} as well as a time limit of 2 hours. Again, we ran the case studies five times and calculated the mean CPU time. The Carnot Battery, without using the tightening based on subgradient heuristic, exceeded the time limit and is therefore not considered in the following.

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). Since the optimization uses only a single MLP with two hidden layers, tightening based on subgradients has no impact on the number of B&B iterations and is almost negligible in terms of CPU time. For the compressor case study without using the tightening based on subgradients, both envelopes require the same amount of B&B iterations, with the envelope of single neuron achieves slightly better CPU times. However, when tightening based on subgradients is applied, the envelope of a neuron performed slightly worse than the envelope of the activation function



(a) (b)

Fig. 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), both without tightening based on subgradients (solid fill) and with tightening based on subgradients (hatched pattern). The Carnot battery case study without tightening based on subgradients exceeds the time limit, and therefore, it is not shown in this figure. Which method works best depends on the case study

regarding CPU time and required more B&B iterations, with a relative difference of approximately 26 % of B&B iterations. In this case study, tightening based on subgradients can impact the MLPs with three hidden layers, reducing iterations and CPU time. Furthermore, for the Carnot battery with tightening based on subgradients, the envelope of a neuron needed 7.2 % more B&B iterations as well as 9.1 % more CPU compared to using the envelope of the activation function. In conclusion, while the envelope of a neuron can be advantageous for small MLPs, the envelope of the activation function combined with tightening based on subgradients outperformed the envelope of neuron for the present case studies.

3 Conclusion and outlook

We implemented the envelope proposed by Carrasco and Muñoz [11] for neurons with hyperbolic tangent and SELU as 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 and CPU time in the context of deeper neural networks.

Furthermore, we demonstrate significant speedups with our subgradient-based tightening from [15]. This tightening yields tighter relaxations and lower CPU time compared to the relaxations without it. However, we observe 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.

Overall conclusion is thus that using separate envelopes of pre-activation and activation function with tightening based on subgradients can lead to tighter relaxations of MLPs than relying solely on envelopes of neurons. Therefore, developing a method that incorporates this tightening of the pre-activation function into the relaxation of the envelope of a neuron could lead to tighter relaxations and improve computational runtime. Additionally, it is also conceivable to develop custom relaxations over multiple layers or even entire MLPs.

Supplementary information. The SI supporting the results presented in this short communication is available as a separate file.

Declarations

Competing interests: The authors declare that they have no conflict of interest.
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.
Data availability: The trained ANNs, the corresponding optimization problems and used scripts are available over this permalink:
http://permalink.avt.rwth-aachen.de/?id=631600
Code availability: The open-source software used in this work is available at:

1. MAiNGO: https://git.rwth-aachen.de/avt-svt/public/maingo
2. MeLOn toolbox: https://git.rwth-aachen.de/avt-svt/public/melon
3. Fork of MC++ used to compute the envelope of a single neuron: https://git.rwth-aachen.de/avt-svt/public/thirdparty/mcpp

References

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

[2] Schweidtmann, A.M., Mitsos, A.: Deterministic global optimization with artificial neural networks embedded. J. Optimiz. Theory App. **180**(3), 925–948 (2019)

[3] Ceccon, F., Jalving, J., Haddad, J., Thebelt, A., Tsay, C., Laird, C.D., Misener, R.: Omlt: Optimization & machine learning toolkit. J. Mach. Learn. Res. **23**(349), 1–8 (2022)

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

[5] Tsoukalas, A., Mitsos, A.: Multivariate McCormick relaxations. J. Global Optim. **59**(2-3), 633–662 (2014)

[6] Smith, E.M.B., Pantelides, C.C.: Global optimisation of nonconvex MINLPs. Comput. Chem. Eng. **21**, 791–796 (1997)

[7] Kearfott, R.B., Du, K.S.: The cluster problem in global optimization: The univariate case. Computing (Suppl.) **9**, 117–127 (1992)

[8] Bishop, C.M.: Pattern Recognition and Machine Learning, 13. (corrected at 8th printing 2009) edn. Information science and statistics. Springer, New York (2009)

[9] Wilhelm, M.E., Wang, C., Stuber, M.D.: Convex and concave envelopes of artificial neural network activation functions for deterministic global optimization. J. Global Optim. **85**(3), 569–594 (2023)

[10] Tjandraatmadja, C., Anderson, R., Huchette, J., Ma, W., Patel, K., Vielma, J.P.: The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. In: Advances in Neural Information Processing Systems, vol. 33, pp. 21675–21686 (2020)

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

[12] Chachuat, B., Houska, B., Paulen, R., Peri'c, N., Rajyaguru, J., Villanueva, M.E.: Set-theoretic approaches in analysis, estimation and control of nonlinear systems. IFAC-PapersOnLine **48**(8), 981–995 (2015). https://github.com/omega-icl/mcpp

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

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

[15] Najman, J., Mitsos, A.: Tighter McCormick relaxations through subgradient propagation. J. Global Optim. **75**(3), 565–593 (2019)

[16] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan,

V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org (2015). https://www.tensorflow.org/

[17] Najman, J., Bongartz, D., Mitsos, A.: Linearization of McCormick relaxations and hybridization with the auxiliary variable method. J. Global Optim. **80**(4), 731–756 (2021)

[18] Lüthje, J.T., Langiu, M., Mitsos, A.: Working Fluid Screening for ORC-Based Carnot Batteries by Deterministic Global Optimization of Design and Nominal Operation. Optim. Eng. (2025)