Convex and Concave Envelopes of Artificial Neural Network Activation Functions for Deterministic Global Optimization[†]

Matthew E. Wilhelm, Chenyu Wang and Matthew D. Stuber*

Process Systems and Operations Research Laboratory, Department of Chemical & Biomolecular Engineering, University of Connecticut, 191 Auditorium Road, Unit 3222, Storrs, 06269, CT, USA.

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

Contributing authors: matthew.wilhelm@uconn.edu; chenyu.wang@uconn.edu;

Abstract

In this work, we present general methods to construct convex/concave relaxations of the activation functions that are commonly chosen for artificial neural networks (ANNs). The choice of these functions is often informed by both broader modeling considerations balanced with a need for high computational performance. The direct application of factorable programming techniques to compute bounds and convex/concave relaxations of such functions often lead to weak enclosures due to the dependency problem. Moreover, the piecewise formulation that defines several popular activation functions, prevents the computation of convex/concave relaxations as they violate the factorable function requirement. To improve the performance of relaxations of ANNs for deterministic global optimization applications, this study presents the development of a library of envelopes of the thoroughly studied rectifier-type and sigmoid activation functions, in addition to the novel self-gated sigmoid-weighted linear unit (SiLU) and Gaussian error linear unit (GELU) activation functions. We demonstrate

[†]Author's final accepted version. Published version: Wilhelm, M.E., Wang, C., and M.D. Stuber. Convex and Concave Envelopes of Artificial Neural Network Activation Functions for Deterministic Global Optimization. *Journal of Global Optimization.* (2022). doi:10.1007/s10898-022-01228-x

that the envelopes of activation functions directly lead to tighter relaxations of ANNs on their input domain. In turn, these improvements translate to a dramatic reduction in CPU runtime required for solving optimization problems involving ANN models to epsilon-global optimality. We further demonstrate that the factorable programming approach leads to superior computational performance over alternative state-of-the-art approaches.

Keywords: artificial neural networks, machine learning, deterministic global optimization, factorable programming, McCormick relaxations, envelopes, Julia programming

1 Introduction

Machine learning and general surrogate modeling approaches provide a means to describe physical phenomena when accurate first-principles models (FPMs) may lead to intractable formulations and when field-specific knowledge may not be adequate to formulate accurate FPMs [1]. These approaches make use of either real-world data or computationally generated datasets to train data-driven models (DDMs) that adequately approximate the underlying system behavior. In many cases, the DDMs primarily serve to reduce intractable models into forms that allow for subsequent analysis, such as process design, sensitivity analysis, or an assessment of controllability. Optimization methods are often embedded in each of these tasks, making the deterministic optimization of nonconvex models, which embed these DDMs, a pursuit of interest.

Numerous data-driven modeling approaches have been applied to engineered systems that include: artificial neural networks (ANNs) [2, 3], Gaussian (Kriging) process models [4–6], and support vector machines [7]. ANNs, in particular, have seen a greatly increased usage with the advent of widely-accessible and highly-capable software tools, such as Tensorflow [8] and Pytorch [9]. These universal approximators represent one class of DDMs that has seen an abundance of usage in recent decades with optimization-based applications ranging from synthesis of biodiesel processes [10], selection of optimal fermentation media [11], optimal control of pressure swing absorption processes [12], design of cross-flow filtration systems [13], and many others [14–16]. More recently, there has been an emergence of interest in applying deep ANNs (usually defined as ANNs with four or more hidden layers in contrast to shallow ANNs that have a single hidden layer) to process system engineering applications as well as standard classification and ranking tasks. One recent example consists of the use of *deep* ANNs to predict multiphase flow characteristics in a pipe [17]. Renewed interest in deep ANNs has resulted in the exploration of novel ANN structures that may provide better performing models (lower computational cost, improved robustness, and better predictive value). Associated with these investigations are several efforts [18–21] to uncover superior activation functions to be used in the more complex structures

that characterize deep ANNs that have the potential to reduce computational time and improve robustness relative to the state-of-the-art ReLU activation function.

Approaches that solve optimization problems deterministically with trained ANNs embedded have largely been limited to ReLU network-based models. Full-space formulations may exploit the equivalency of ReLU networks to mixed-integer linear programs (MILP) [22–25] or adapt ReLU network representations of piecewise linear functions to perform adaptive partitioning and domain tightening [26]. The resulting problems are then solved using state-of-the-art MILP (or MINLP solvers if nonlinear terms are present). Despite recent developments enabling deterministic global optimization of certain ANNs for process systems engineering applications [3, 27], there still remains a need for theoretical developments that enable support for a broader library of activation functions and additional families of ANNs. Namely, trained neural networks incorporating these activation functions may be described by systems of equations and, in turn, embedded in a mathematical program. Unfortunately, most activation functions embedded in this manner are nonlinear and exhibit significant nonconvexities that lead to difficulties in solving the resulting optimization formulation.

Explicit consideration of activation functions may lead to tighter relaxations of nonconvex optimization problems involving ANNs. The availability of tighter relaxations remains desirable as their use may greatly accelerate the convergence of the branch-and-bound (B&B) algorithm that underlies all commercially available state-of-the-art deterministic global optimizers. Support for more general ANNs is also desirable for two other important reasons. First, it increases the variety of ANN model forms that may be developed for use in global optimization applications. Second, it allows for the integration of a broader family of legacy models built for general predictive purposes. These may be difficult to adapt to alternative surrogate model formulations due to domain-specific modeling considerations or potential logistical hurdles such as the unavailability of legacy training data. As such, methods that are directly applicable to these models are desirable. In this paper, we make the following novel contributions that serve to address these outstanding issues:

1. We discuss ANN structures of current research interest and reduced-space reformulations for specialized ANN structures that may participate in global optimization formulations (the reader is directed to Section 4 for a summary of reduced-space formulations). In particular, we highlight a collection of activation functions without standard factorable representations using software libraries and categorize these according to convexity properties.
2. We derive novel convex/concave envelopes for the increasingly popular implicitly regularizing activation functions sigmoid-weighted linear unit (SiLU) and Gaussian error linear unit (GELU).
3. We analyze the convexity properties of numerous common activation functions and highlight how naïve McCormick relaxations lead to overestimation

of convex/concave relaxations due to the dependency problem (i.e., overestimation inherent to set-valued arithmetic due to the inability to recognize multiple occurrences of the same variable in a given expression [28, 29]).
4. We illustrate that the use of envelopes for common/popular activation functions leads to increased performance relative to a naïve application of composition rules originating from Garth McCormick's foundational work [30] using a randomly generated benchmark set.

These contributions ultimately lead to faster solution times associated with reduced-space optimization methods for a wide variety of ANN-based machine learning models. Moreover, our contributions allow for an extended library of activation functions to be utilized in nonlinear programs (NLPs) that must be solved with a certificate of global optimality.

In this paper, we present new developments on the relaxation of activation functions common in recent ANN-based models. In Section 2, we detail the mathematical conventions used in the paper while briefly reviewing relevant results in convex analysis and machine learning. Subsequently, in Section 3, we develop and analyze convex and concave relaxations of several activation functions that have become increasingly prevalent in broader machine learning applications. In Section 4, we describe full-space and reduced-space formulations for global optimization problems; we then proceed to detail how ANNs may readily be incorporated into either formulation. In Section 5, we present the numerical results arising from a randomly generated benchmark set that illustrate the performance improvements readily achievable using these novel envelopes. Lastly, in Section 6, we reflect on current technical challenges and suggest future directions for subsequent research.

2 Mathematical Background

2.1 Interval Arithmetic

Scalar quantities are represented as lower-case letters (e.g., a) and vectors are denoted by boldface lower-case letters (e.g., $\mathbf{a}$). Let $A = [\mathbf{a}^L, \mathbf{a}^U]$ represent an n-dimensional real interval vector that is a nonempty compact set defined as $A = \{\mathbf{a} \in \mathbb{R}^n : \mathbf{a}^L \leq \mathbf{a} \leq \mathbf{a}^U\}$ with $\mathbf{a}^L$ and $\mathbf{a}^U$ the lower and upper bounds of the interval, respectively, and A_i representing the i-th component of the vector A. Additionally, let $\mathbb{IR}^n$ be the set of all n-dimensional real intervals and for any $D \subset \mathbb{R}^n$, $\mathbb{I}D = \{X \in \mathbb{IR}^n : X \subset D\}$ is the set of all interval subsets of D. A set B^n is defined as the Cartesian product $B^n \equiv B \times B \times \cdots \times B$ for $B \subset \mathbb{R}$. Further, let the *diameter* of a scalar-valued interval, X, be defined as $\mathrm{diam}(X) = x^U - x^L$ and the *radius* be given by $\mathrm{rad}(X) = \mathrm{diam}(X)/2$.

2.2 Convex and Concave Relaxations

Definition 1 (Univariate Intrinsic Function [31])**.** The function $u : B \subset \mathbb{R} \to \mathbb{R}$ is a *univariate intrinsic function* if, for any $A \in \mathbb{I}B$, the following are known and can be evaluated computationally:

1. an interval extension of u on A that is an inclusion function of u on A,
2. a concave relaxation of u on A,
3. a convex relaxation of u on A.

Definition 2 (Factorable Function [31])**.** A function $\mathscr{F} : Z \subset \mathbb{R}^n \to \mathbb{R}$ is *factorable* if it can be expressed in terms of a finite number of factors $v_1, \ldots, v_m$, such that given $\mathbf{z} \in Z$, $v_i = z_i$ for $i = 1, \ldots, n$, and v_k is defined for $n \leq k \leq m$ as either

1. $v_k = v_i + v_j$, with, $i, j < k$, or
2. $v_k = v_i v_j$, with, $i, j < k$, or
3. $v_k = u_k(v_i)$, with, $i < k$, where $u_k : B_k \to \mathbb{R}$ is a univariate intrinsic function,

and $\mathscr{F}(\mathbf{z}) = v_m(\mathbf{z})$, for every $\mathbf{z} \in Z$. A vector-valued function is factorable if each of its components are factorable functions.

Definition 3 (Cumulative Mapping [31])**.** Let the *cumulative mapping* v_k be the mapping $v_k : Z \to \mathbb{R}$ defined for each $\mathbf{z} \in Z$ by the value $v_k(\mathbf{z})$ when the factors of $\mathscr{F}$ are computed recursively, as per Definition 2, beginning from $\mathbf{z}$.

Definition 4 (Convex and Concave Relaxations [32])**.** Given a convex set $Z \subset \mathbb{R}^n$ and a function $f : Z \to \mathbb{R}$, a convex function $f^{cv} : Z \to \mathbb{R}$ is a *convex relaxation* of f on Z if $f^{cv}(\mathbf{z}) \leq f(\mathbf{z})$ for every $\mathbf{z} \in Z$. A concave function $f^{cc} : Z \to \mathbb{R}$ is a *concave relaxation* of f on Z if $f^{cc}(\mathbf{z}) \geq f(\mathbf{z})$ for every $\mathbf{z} \in Z$.

Note that this definition involves scalar functions. However, convex and concave relaxations of vector valued functions $\mathbf{f} : Z \to \mathbb{R}^n$ are defined by applying the above inequalities componentwise [33].

Definition 5 (Convex and Concave Envelope [34])**.** Let $f : S \to \mathbb{R}$ where $S \subset \mathbb{R}^n$ is a nonempty convex set. The *convex envelope* of f on S is the convex relaxation $f^{cv,env} : S \to \mathbb{R}$ such that $f^{cv}(x) \leq f^{cv,env}(x)$ holds for all $x \in S$ and every convex relaxation f^{cv} of f on S. Similarly, the *concave envelope* of f on S is the concave relaxation $f^{cc,env} : S \to \mathbb{R}$ such that $f^{cc}(x) \geq f^{cc,env}(x)$ holds for all $x \in S$ and every concave relaxation f^{cc} of f on S.

Definition 6 (McCormick Relaxation [32])**.** Relaxations of factorable functions that are formed from the recursive application of univariate composition, binary multiplication, and binary addition from convex and concave relaxations of univariate intrinsic functions, without the introduction of auxiliary variables, are referred to as *McCormick relaxations*.

Proposition 1 (Univariate McCormick Composition Rule [32])**.** Let $Z \subset \mathbb{R}^n$ and $X \subset \mathbb{R}$ be nonempty convex sets. Consider the composite function $w = \phi \circ q$ where $w : Z \to \mathbb{R}$ is continuous, $\phi : X \to \mathbb{R}$, let $q(Z) \subset X$. Let $q^{cv} : Z \to \mathbb{R}$ and $q^{cc} : Z \to \mathbb{R}$ be convex and concave relaxations of q on Z, respectively. Let $\phi^{cv} : X \to \mathbb{R}$ and $\phi^{cc} : X \to \mathbb{R}$ be convex and concave relaxations of ϕ on X, respectively. Let $\xi^*_{\min} \in X$ be a point at which ϕ^{cv} attains its infimum on X and let $\xi^*_{\max} \in X$ be a point at which ϕ^{cc} attains its supremum on X. Then the convex and concave relaxations are, respectively, given by

$$w^{cv} : Z \to \mathbb{R} : \mathbf{z} \mapsto \phi^{cv}(\text{mid}(q^{cv}(\mathbf{z}), q^{cc}(\mathbf{z}), \xi^*_{\min})) \tag{1}$$

$$w^{cc} : Z \to \mathbb{R} : \mathbf{z} \mapsto \phi^{cc}(\text{mid}(q^{cv}(\mathbf{z}), q^{cc}(\mathbf{z}), \xi^*_{\max})). \tag{2}$$

In the above, the $\text{mid}(\cdot,\cdot,\cdot)$ function takes the median value of its three arguments. Generally, the application of Proposition 1 requires the use of closed-form expressions, which are available for all standard operations (e.g. +, ×, exp, log), to determine the values of $\xi^*_{\max}$ and $\xi^*_{\min}$ in conjunction with defined forms of the relaxations ϕ^{cv} and ϕ^{cc}. Convex and concave envelopes of univariate intrinsic functions (such as exp and tanh) are available in existing relaxation libraries (e.g. [35]) and are used throughout this paper for calculations unless otherwise specified. In the case of "×", the rules presented in [32] are used for the calculation of relaxations and the max operator is addressed using the standard reformulation $\max(x, y) = (x + y + |x - y|)/2$.

2.3 Artificial Neural Networks

One of the most common ANN structures is that of the multilayer perceptron (MLP). The MLP is a class of feedforward ANNs that consists of a directed acyclic graph (DAG) containing n layers enumerated $k = 1, \ldots, n$. The first layer consists of inputs to the MLP. The subsequent $k = 2, \ldots, n-1$ layers are the *hidden layers*, with $k = n$ the *output layer*. The number of neurons in layer k is denoted $m^{(k)}$. Let $\mathbf{a}^{(k)} \in \mathbb{R}^{m^{(k)}}$ be the output vector of layer k. Accordingly, $\mathbf{a}^{(1)}$ is the input vector and $\mathbf{a}^{(n)}$ is the output vector of the MLP. For layers $k \in \{2, \ldots, n\}$, the vector $\mathbf{a}^{(k)}$ is defined componentwise by

$$a_i^{(k)} = f^{(k)}\left(\left(\mathbf{w}_i^{(k-1)}\right)^{\mathrm{T}} \mathbf{a}^{(k-1)} + b_i^{(k-1)}\right), \; i = 1, \ldots, m^{(k)}, \tag{3}$$

where $f^{(k)} : \mathbb{R} \to \mathbb{R}$ are activation functions, $\mathbf{W}^{(k-1)} = \left[\mathbf{w}_1^{(k-1)} \; \mathbf{w}_2^{(k-1)} \; \cdots \; \mathbf{w}_{m^{(k)}}^{(k-1)}\right] \in \mathbb{R}^{m^{(k)} \times m^{(k-1)}}$ is a *weight matrix*, and $\mathbf{b}^{(k-1)} \in \mathbb{R}^{m^{(k)}}$ is a *bias vector*. For ease of introduction, we define $\mathbf{o} : \mathbb{R}^{m^{(1)}} \to \mathbb{R}^{m^{(n)}}$ to represent the input-output function for a generic DDM. In the case of an MLP, we have $\mathbf{a}^{(n)} = \mathbf{o}(\mathbf{a}^{(1)})$. A depiction of this type of network is provided in Figure 1. When a feedforward ANN is trained, the weight matrices and bias vectors become optimization variables, while the values of the

input vector $a_i^{(1)}$ for $i = 1,\ldots,m^{(1)}$, are treated as parameters. When a trained feedforward ANN is embedded in an optimization problem, the weight matrices and bias vectors are fixed to constant parameter values.

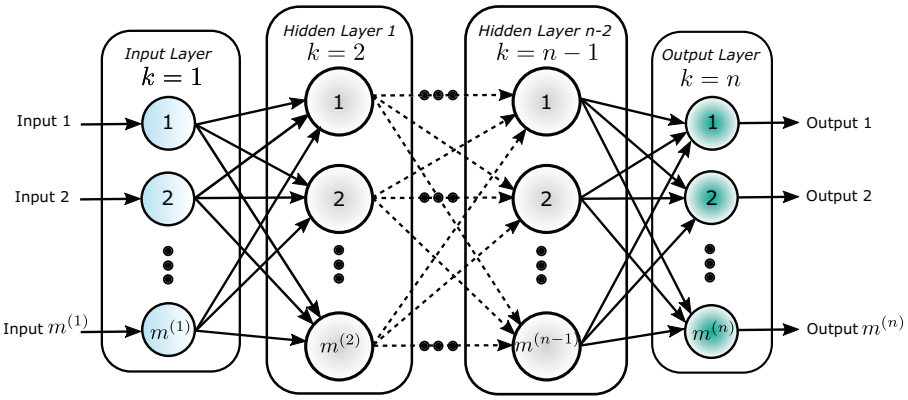

Fig. 1 The directed acyclic graph representation of a multilayer perceptron with n layers. The input and output layers are the first ($k = 1$) and n-th ($k = n$) layers, respectively. The hidden layers are labeled $k = 2$ to $k = n - 1$. Note that the multilayer perceptron is a fully-connected network in which each neuron in layer $k - 1$ is connected to all neurons in the subsequent layer k.

It should be noted that while the MLP structure may be easily decomposed into a factorable representation, many ANNs of active interest may not. For example, residual networks and recurrent neural networks have analogous continuous-time representations [36–39]. These continuous-time representations are often desirable as they may be evaluated using state-of-the-art ODE integrators, and in turn circumvent the need to search for an optimal number of hidden layers. This has motivated recent interest in neural-ordinary differential equations [40, 41]. These models can be addressed by exploiting specialized continuous-time relaxation methods for ODEs [42–45]. Alternatively, deep ANNs with an implicit representation [46] may require the solution of nonlinear systems via fixed-point methods to evaluate the neural network outputs and specialized relaxation methods for fixed-point methods need to be applied [33]. However, in both cases, using tighter relaxations of the activation functions participating in the overall network will result in tighter relaxations of the overall network. Nonetheless, the methods described herein will be applicable to generalized feedforward neural networks [47], including deep feedforward networks [48], extreme learning machines [49], discrete recurrent neural networks [50], and deep residual networks [51] as activation functions represent a key component of each of these networks.

3 Relaxations of Activation Functions

To solve NLPs with a certificate of global optimality, the use of a deterministic global optimization method (e.g., B&B) is required [52]. These methods typically require that convex (and concave) relaxations of the participating nonconvex expressions may be readily computed. After the seminal work of McCormick [30], there has been a significant effort to develop libraries of relaxations of common mathematical expressions and intrinsic functions that are often encountered in common mathematical models and relevant optimization problems. In this section, we further contribute to these efforts by developing convex and concave relaxations of common activation functions encountered in ANNs.

Previous work has focused on ANNs with a hyperbolic tangent [3, 27, 53] activation function. It was noted that increasing ANN depth is undesirable when holding the number of neurons fixed, due to the overestimation of relaxations [3]. In this section, we review relaxations of several common activation functions that have been developed more recently, with special attention paid to the Gaussian error linear unit (GELU) and the sigmoid-weighted linear unit (SiLU) functions. Many of these activation functions are known to perform better in a deep network configuration, specifically by either reducing the training time for equivalent network structures or by enabling networks with fewer terms that yield a more accurate fit to the data. As a consequence of using these activation functions, the number of nonconvex expressions that participate in a DDM may be reduced while maintaining the desired accuracy. This can lead to more tractable formulations of otherwise prohibitively expensive optimization problems.

We also note that the construction of relaxations of activation functions using standard McCormick libraries (e.g., [35, 54]) may often be inadequate. For example, in some cases, the piecewise definitions cannot be readily implemented via overloading approaches, and in other cases, the direct application of overloading approaches leads to weak relaxations. Note that this applicability to broader classes of ANNs is desirable, as it allows legacy models, built for long-term prediction, to be readily embedded in process simulations for optimization-based design tasks. We begin this section with a discussion of common families of activation functions and their associated convex/concave envelopes. Finally, we derive the envelopes for the recent SiLU and GELU activation functions that will be utilized in the case studies in this paper.

3.1 Convex Activation Functions

One of the most ubiquitous activation functions currently used in machine learning is that of the Rectified Linear Unit (ReLU), $f : \mathbb{R} \to \mathbb{R} : x \mapsto \max(x, 0)$. The development of the ReLU represented a significant breakthrough in the supervised training of deep ANNs [55], and has since become the standard activation function used in deep learning. The simplicity of the ReLU in concert with the lack of any vanishing gradient issue, wherein nonlinearities lead to near-singular values of the input-output Jacobian, have often been noted as distinct advantages of

this activation function. Additionally, networks comprised of ReLU functions may be readily modeled as MILPs [22] and solved using well-established MILP solvers. One proposed alteration to the ReLU resulted in the development of the exponential linear unit (ELU) which allows for negative outputs and avoids the "dying" ReLU problem — when ReLU neurons participating in a network only output 0 for any input during network training [56] — at the cost of significantly more complex arithmetic operations [57]. A scaled ELU was also proposed that avoids both vanishing and exploding gradient problems and incorporates an internal normalization routine [58]. Many of these newly proposed activation functions belong to a family of monotonically increasing convex functions.

The convex envelope of a univariate scalar-valued convex function f on a compact set $[a,b]$ is simply the function itself. Whereas, its concave envelope is the affine function that joins the end points $f(a)$ and $f(b)$ [30]. Relaxations of compositions involving these terms can be computed using the Univariate McCormick Composition Theorem, Proposition 1 herein. While some activation functions such as ReLU have exact convex/concave relaxations when computed via naïve McCormick composition [30, 32], others are defined as compositions of multiple algebraic expressions that lead to overestimation due to the well-known dependency problem. Functions such as ReLU consist of only one expression and are already included in McCormick relaxation software libraries [35, 54, 59]. Other activation functions such as Maxsig, Maxtanh, and Softplus have relaxations that are weaker than their envelopes when computed using the algebraic expressions listed in Table 1, as illustrated in Figure 2, due to the dependency problem associated with the computation of relaxations of composite functions. Other convex activation functions, namely, ELU, SELU, and parametric ReLU, cannot be easily implemented using the frameworks of [30] and [32], as the conditional statement in the activation function definitions breaks the factorable function assumption inherent to these relaxation algorithms.

Two other considerations serve to motivate interest in alternatives to the ReLU function. First, the use of a twice-differentiable activation function is desirable, as twice-differentiability is generally sufficient to ensure second-order pointwise convergence of relaxations of ANNs [60]; a key consideration in mitigating the clustering problem present in deterministic global optimization algorithms [61]. Although no function other than softplus listed in Table 1 is twice differentiable, one can reasonably expect a more significant degree of nondifferentiability to generally occur within nonsmooth activation functions as the nonsmooth behavior may arise from both the mid$(\cdot)$ operator present in the McCormick composition rule and the envelope itself. Second, the time spent calculating relaxations of the activation function may be minor when compared with the time spent in solving linear, mixed-integer, and convex nonlinear problems in any given iteration of the B&B algorithm. As such, it may be beneficial to compute tighter relaxations of slightly more computationally expensive terms if one may substantially reduce the overall number of iterations performed by the global solver and gain additional benefits associated with bounds tightening algorithms or other key heuristics

Activation Function	Form $f(x)$	Derivative $f'(x)$	Source
ReLU	$\max\{0, x\}$	$\begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$	[64]
Parametric ReLU $0 < \alpha < 1$	$\begin{cases} x, & \text{if } x > 0 \\ \alpha x, & \text{otherwise} \end{cases}$	$\begin{cases} 1, & x > 0 \\ \alpha, & x \leq 0 \end{cases}$	[65]
Maxsig (Let a be a root of $1 + \exp(-a) - a^{-1}$)	$\max(x, 1/(1+\exp(-x)))$	$\begin{cases} 1, & x > a \\ \exp(-x)/(\exp(-x)+1)^2, & x \leq a \end{cases}$	[66]
Maxtanh	$\max(x, \tanh(x))$	$\begin{cases} 1, & x > 0 \\ \operatorname{sech}^2(x), & x \leq 0 \end{cases}$	[66]
Softplus	$\log(1+\exp(x))$	$(1+\exp(-x))^{-1}$	[67]
Exponential Linear Unit (ELU) $\alpha > 0$	$\begin{cases} x, & x > 0 \\ \alpha(\exp(x)-1), & x \leq 0 \end{cases}$	$\begin{cases} 1, & x > 0 \\ \alpha \exp(x), & x \leq 0 \end{cases}$	[57]
Scaled Exponential Linear Unit (SELU) $\lambda = 1.0507$, $\alpha = 1.67326$	$\lambda \begin{cases} x, & x > 0 \\ \alpha(\exp(x)-1), & x \leq 0 \end{cases}$	$\begin{cases} \lambda, & x > 0 \\ \lambda\alpha \exp(x), & x \leq 0 \end{cases}$	[58]

Table 1 Convex activation functions and their first derivatives are defined in this table.

[62, 63]. These considerations remain important for sigmoidal (convexoconcave) activation functions and novel self-gating activation functions.

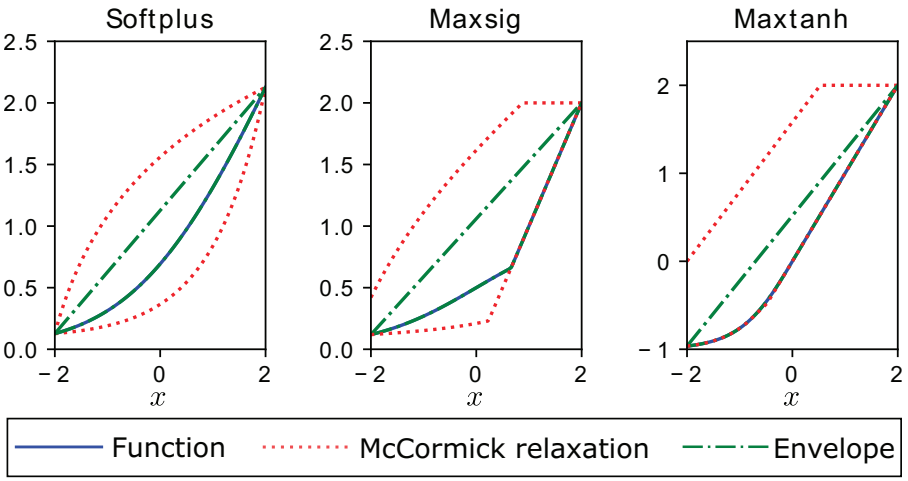

Fig. 2 Relaxations of some common rectifier-like activation functions are weaker than their corresponding convex/concave envelopes when the relaxations are computed using the rules presented in [30]. The activation function, the relaxations of form $f(x)$ from Table 1 computed according to the rules presented in [30], and convex/concave envelopes of the activation function, are shown in each subplot on the domain $x \in [-2, 2]$. These plots illustrate the overestimation of the classical McCormick relaxation approach compared to the envelopes for (**Left**) Softplus, (**Middle**) Maxsig, and (**Right**) Maxtanh functions.

Activation Function	Form $f(x)$	Derivative $f'(x)$
Softsign	$x/(1+\mathrm{x})$	$(1+\mathrm{x})^{-2}$
Hyperbolic tangent	$\tanh(x)$	$\mathrm{sech}^2(x)$
Penalized hyperbolic tangent	$\begin{cases}\tanh(x), & x>0\\ \tanh(\alpha x), & x\leq 0\end{cases}$	$\begin{cases}\mathrm{sech}^2(x), & x>0\\ \alpha(\mathrm{sech}^2(\alpha x)), & x\leq 0\end{cases}$
Sigmoid	$(1+\exp(-x))^{-1}$	$\exp(-x)(1+\exp(-x))^{-2}$
Bipolar sigmoid	$(1-\exp(-x))/(1+\exp(-x))$	$2\exp(x)(1+\exp(x))^{-2}$

Table 2 Convexoconcave activation functions and their first derivatives are defined in this table [66, 68].

3.2 Convexoconcave Activation Functions

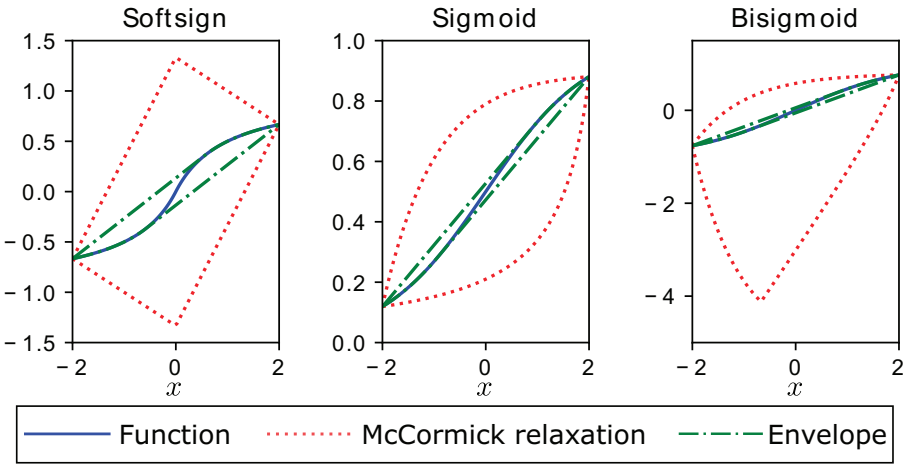

Fig. 3 Many sigmoidal activation functions have relaxations that are weaker than their envelopes when computed using the rules presented in [30]. The activation function, the relaxations of form $f(x)$ from Table 2 computed according to the rules presented in [30], and convex/concave envelopes of the activation function are shown in each subplot on the domain $x \in [-2,2]$. These plots illustrate the overestimation of the classical McCormick relaxation approach compared to the envelopes for (**Left**) Softsign, (**Middle**) Sigmoid, and (**Right**) Bisigmoid functions.

Some of the earliest-used activation functions for ANNs are convexoconcave — a univariate function consisting of a convex region followed by a concave region. The preliminary use cases for convexoconcave activation functions primarily involve sigmoid and hyperbolic tangent activation functions [69] participating in shallow ANNs used for regression or classification tasks. As the sigmoid activation functions are bounded, the inclusion of sigmoid layers may be used to constrain the range of ANN predictions. Continued investigations into sigmoid-shaped ANNs have focused on reducing the necessary computational time and minimizing the vanishing gradient problem, leading to forms such as Softsign (i.e.,

ElliotSig) [48, 70]. Several equivalent algebraic forms of the hyperbolic tangent were examined by [3]. The authors noted that the direct application of McCormick composition rules leads to substantially weaker relaxations than the envelope for the majority of alternative algebraic forms. We demonstrate here that these results hold for many other convexoconcave activation functions. While astute usage of algebraic rearrangements may improve the quality of relaxations, such as the conversion of the bipolar sigmoid function to the equivalent $\tanh(x/2)$ form, this is not possible for all activation functions.

Convex/concave envelopes of convexoconcave activation functions can be computed using the rules described by [30] and [71]. A tie point x_m^{cv} is computed at which the function's derivative (provided that the function is differentiable at x_m^{cv}) equals the slope of the secant line between $(f(x_m^{cv}), x_m^{cv})$ and $(f(x^U), x^U)$. Similarly, a tie point x_m^{cc} is computed at which the function's derivative (provided the function is differentiable at x_m^{cc}) equals the slope of the secant line between $(f(x_m^{cc}), x_m^{cc})$ and $(f(x^L), x^L)$. That is,

$$f^{cv,env}(x) = \begin{cases} f(x^U) + \frac{f(x^U)-f(x_m^{cv})}{x^U - x_m^{cv}}(x - x^U), & x \geq x_m^{cv} \\ f(x), & \text{otherwise} \end{cases} \tag{4}$$

$$f^{cc,env}(x) = \begin{cases} f(x^L) + \frac{f(x^L)-f(x_m^{cc})}{x^L - x_m^{cc}}(x - x^L), & x \leq x_m^{cc} \\ f(x) & \text{otherwise.} \end{cases} \tag{5}$$

As illustrated in Figure 3, the envelopes yield substantially tighter relaxations than the direct application of McCormick composition rules [32] to the arithmetic expressions presented in Table 2.

3.3 Other Activation Functions

Models involving the ReLU function, as well as many of the convex activation functions presented here, lead to nondifferentiability, which may present issues for subsequent optimization and analysis. Moreover, sigmoid activation functions often suffer from a vanishing gradient issue when applied in deep ANNs. Two other activation functions that have recently garnered significant interest are differentiable and avoid the vanishing gradient problem. The GELU function, denoted as $\text{gelu} : \mathbb{R} \to \mathbb{R}$, was developed as a means of merging stochastic dropout and zoneout procedures naïvely with the ReLU activation function [72]. The cumulative distribution function arising from the stochastic zero or identity transformation yields the $\text{gelu}(\cdot)$ function, defined as:

$$\text{gelu}(x) = \frac{x}{2}\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right). \tag{6}$$

Another activation function of particular interest in the past few years that was originally developed for use in reinforcement learning by [73], is the SiLU function,

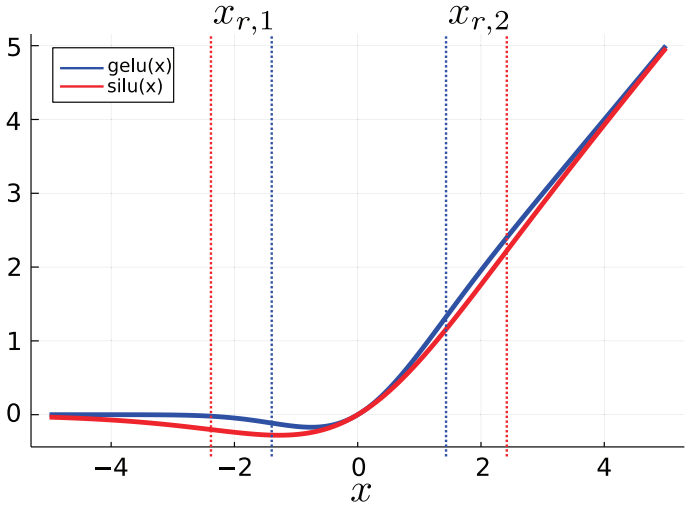

Fig. 4 The gelu(·) and silu(·) activation functions are plotted on the domain $X = [-5,5]$. The left and right inflection points, $x_{r,1}$ and $x_{r,2}$, respectively, are marked by the dashed vertical lines with the color corresponding to the respective function. Each function is concave on the domain $(-\infty, x_{r,1}]$, convex on the domain $[x_{r,1}, x_{r,2}]$, and concave on the domain $[x_{r,2}, +\infty)$.

herein denoted silu : $\mathbb{R} \to \mathbb{R}$, and defined as:

$$\text{silu}(x) = \frac{x}{1 + \exp(-x)}. \tag{7}$$

It was noted that the silu(·) function exhibits a basic self-stabilizing property [74]. Similar to the gelu(·) function, the global minimum of the silu(·) function serves as an implicit regularizer that inhibits learning of large magnitude weights. Subsequent exploratory work by [75] used a reinforcement learning approach with a recurrent ANN to identify several potentially useful activation functions. They found the silu(·) activation function and provided strong numerical evidence that it outperforms a myriad of alternative activation functions.

Figure 4 illustrates the gelu(·) and silu(·) activation functions and their respective inflection points. We note that the convexity of silu(·) parallels that of gelu(·). Namely, each function has inflection points $x_{r,1}$ and $x_{r,2}$ that bound a region where they are convex. Outside this region, the functions are concave. These inflection points occur at $\pm\sqrt{2}$ and approximately ± 2.39935 for gelu(·) and silu(·), respectively. Given a domain X, if $x_{r,1}, x_{r_2} \notin X$, then the function is either concave or convex on X and the envelope may be constructed as described in Section 3.1. If $x_{r,1} \notin X, x_{r,2} \in X$, then $f \in \{\text{gelu}, \text{silu}\}$ is convexoconcave on X and convex and concave envelopes may be computed from (4) and (5), respectively. If $x_{r,1} \in X, x_{r,2} \notin X$, then $g = -f$, with $f \in \{\text{gelu}, \text{silu}\}$, is convexoconcave and its envelope may be computed using (4) and (5), and then by applying the identity $(f^{cv,env}, f^{cc,env}) = (-g^{cc,env} - g^{cv,env})$. We now proceed to derive the convex and concave envelopes of gelu(·) and silu(·) on X, with $x_{r,1}, x_{r,2} \in X$ in the following Theorem 1.

Theorem 1. (Convex/Concave Envelopes of gelu(·), silu(·)) Let $f : \mathbb{R} \to \mathbb{R}$ be defined as either gelu(·) or silu(·), as defined in (6) and (7), respectively. Let $x_{r,1}, x_{r,2} \in X \in \mathbb{IR}$ be the inflection points of $f(\cdot)$ such that $x_{r,1} < x_{r,2}$ and let $x_{\min}$ be the point at which f attains its minimum on $\mathbb{R}$. Let $f^{cv}, f^{cc} : X \to \mathbb{R}$ denote convex and concave relaxations of f on X, respectively. Then, for each $x \in X$, $f^{cv}(x)$ is given by

$$f^{cv}(x) = \begin{cases} f(x^L) + \frac{f(x^{cv}_{m,1}) - f(x^L)}{x^{cv}_{m,1} - x^L}(x - x^L) & x < x^{cv}_{m,1} \\ f(x) & x^{cv}_{m,1} \le x < x^{cv}_{m,2} \\ f(x^{cv}_{m,2}) + \frac{f(x^U) - f(x^{cv}_{m,2})}{x^U - x^{cv}_{m,2}}(x - x^{cv}_{m,2}) & x^{cv}_{m,2} \le x \end{cases} \tag{8}$$

where the tie points $x^{cv}_{m,1} \in [x_{r,1}, x_{\min}]$ and $x^{cv}_{m,2} \in [x_{\min}, x_{r,2}]$ are points that satisfy the following, respectively:

$$f(x^{cv}_{m,1}) - f(x^L) - (x^{cv}_{m,1} - x^L) f'(x^{cv}_{m,1}) = 0, \tag{9}$$

$$f(x^U) - f(x^{cv}_{m,2}) - (x^U - x^{cv}_{m,2}) f'(x^{cv}_{m,2}) = 0. \tag{10}$$

Similarly, for each $x \in X$, $f^{cc}(x)$ is given by

$$f^{cc}(x) = f(x^L) + \frac{f(x^U) - f(x^L)}{x^U - x^L}(x - x^L). \tag{11}$$

Proof Note that under the hypothesis $x_{r,1}, x_{r,2} \in X$ and the convexity/concavity properties of $f(\cdot) \in \{\text{gelu}(\cdot), \text{silu}(\cdot)\}$ on X, the envelopes of f are required on a domain consisting of three adjoining regions where f is concave ($R_1 = [x^L, x_{r,1}]$), convex ($R_2 = [x_{r,1}, x_{r,2}]$), and concave ($R_3 = [x_{r,2}, x^U]$), such that $X = \bigcup_i R_i$. First, we note that both gelu(·) and silu(·) are twice differentiable. Beginning with the convex envelope, we note that f is monotonically decreasing and since $x_{\min} > x_{r,1}$, $f(x) > f(x_{\min})$ for all $x \in R_1$. As a consequence, there exists a tie point $x^{cv}_{m,1} \in [x_{r,1}, x_{\min}]$ that satisfies (9) such that a secant line may be constructed between $(x^L, f(x^L))$ and the point $(x^{cv}_{m,1}, f(x^{cv}_{m,1}))$. This secant line is defined by the first case of (8) and is the convex envelope of f on R_1 and part of R_2. Similarly, since $x_{\min} < x_{r,2}$ and f is monotonically increasing on R_3, $f(x) > f(x_{\min})$ for all $x \in R_3$. Thus, there exists a tie point $x^{cv}_{m,2} \in [x_{\min}, x_{r,2}]$ satisfying (10), such that a secant line may be constructed between $(x^{cv}_{m,2}, f(x^{cv}_{m,2}))$ and the point $(x^U, f(x^U))$. This secant line is defined by the third case of (8) and is the convex envelope of f on R_3 and part of R_2. For $x \in [x^{cv}_{m,1}, x^{cv}_{m,2}]$, $f(x)$ is convex and therefore is trivially its own convex envelope on this subdomain. Thus, (8) defines the convex envelope of f on X. Next, we consider the concave envelope. Since f is monotonically decreasing on $[x^L, x_{\min}]$ and monotonically increasing on $[x_{\min}, x^U]$, the concave envelope is defined by the secant line connecting the endpoints $(x^L, f(x^L))$ and $(x^U, f(x^U))$, defined by (11). □

We now proceed to examine the convergence behavior of activation function envelopes and contrast these with naïve McCormick calculations.

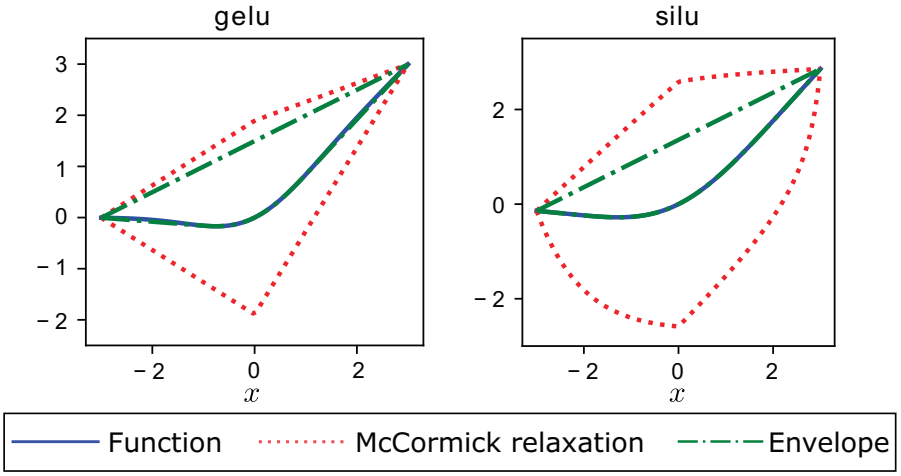

Fig. 5 Left: The Gaussian error linear unit (gelu(·)) function, the convex/concave relaxations of $x\left(1+\text{erf}\left(x/\sqrt{2}\right)\right)/2$ computed according to the rules presented in [30] and the convex/concave envelope of the gelu function are plotted on $x \in [-3,3]$. **Right:** The sigmoid-weighted linear unit (silu(·)) function, the convex/concave relaxations of $x/(1+\exp(-x))$ computed according to the rules presented in [30] and the convex/concave envelope of the silu function are plotted on $x \in [-3,3]$.

3.4 Convergence Properties of Convex/Concave Relaxations of Activation Functions

We now compare the relative performance of naïve McCormick relaxations versus the newly defined envelopes of the library of activation functions: Softplus, Maxsig, Maxtanh, Softsign, Sigmoid, Bisigmoid, SiLU, and GELU. Figure 2, Figure 3, and Figure 5 visualize the relative tightness of McCormick relaxations versus the envelopes. Here, we quantify the performance of the proposed envelopes versus the naïve McCormick relaxations using two analyses. First, the relative computational times are measured for constructing convex and concave relaxations on a domain and evaluating these relaxations and subgradients at a single point. The second analysis compares the relaxations of prototypical ANNs of varying depth under a width metric. Relaxations of each activation function have been implemented in the McCormick.jl [35] subpackage of the author's optimization package EAGO.jl [59], and is openly available.

The computational time comparison was conducted using the BenchmarkTools.jl[76] package in Julia v1.6.2 with the default settings. For each benchmark run, 10^4 samples were used with an automatically-generated number of expression evaluations per sample (chosen by the package for accurate timings). Relaxations of each activation function on the domain $X = [-3,3]$ were computed using the implemented envelopes as well as applying the standard McCormick rules to the algebraic forms of each activation function. This domain was chosen as it encloses the inflection points of all activation functions considered.

Function	Softplus	Maxsig	Maxtanh	Softsign	Sigmoid	SiLU	GELU
envelope (μs)	0.172	0.134	23.1	0.919	0.898	1.41	249
naïve (μs)	0.158	0.384	14.9	0.0199	0.221	0.189	135
τ (%)	8.74	-65.2	54.8	4519	306	643	83.9

Table 3 The costs of calculating convex and concave relaxations and corresponding subgradients of the considered activation functions are tabulated for the newly defined envelopes and naïve McCormick relaxations. The absolute CPU times (μ s) and relative times (%) τ are reported. For almost all activation functions in this table, the envelope calculations are more expensive (and sometimes significantly) due to the necessity of calculating the tie points.

The timing results for calculating envelopes are recorded in Table 3 as percentages relative to naïve McCormick relaxations. In most cases, the envelopes are significantly more computationally expensive to calculate. This is because the tie points x_m^{cv}, x_m^{cc} must be calculated to construct the envelopes, which in turn requires the solution of nonlinear algebraic equations. Softplus and Maxsig are exceptions because they are convex on X, and thus their envelopes are trivial and less expensive to calculate than naïvely applying the McCormick composition rules. However, the increased CPU time required to compute envelopes is still considered small when compared to the CPU time requirements for the other subroutines encountered in global optimization, such as optimization-based bounds tightening, the exact solution of MILPs, and local solution of NLPs, which are all bottlenecks in the solution of global optimization problems. Moreover, it is well-known that the use of tighter relaxations accelerates node fathoming, and therefore, reduces the number of subproblems that must be considered. For complex nested subexpressions, such as ANNs, the gains achieved may be substantial. We demonstrate the tightness conferred to the relaxations of ANNs by the use of envelopes using the illustrative numerical example below.

For the analysis of relaxations of ANNs on an input domain, a simple MLP is generated with two hidden layers, each with a single type of activation function $f^{(k)}(\cdot) \in \{\text{gelu}(\cdot), \text{silu}(\cdot)\}$, and a fully-connected affine layer defining a single output value (i.e., $f_{mlp} = a^{(4)} = o(a^{(1)})$). Ten neurons are included in each layer and the weights and bias values were randomly selected from a uniform distribution between 0 and 1. To simplify the calculations, we set the input variable $a^{(1)}$ to $p \in P = [-\delta p, \delta p]$ and allowed δp to vary by factors of 10 from 10^{-1} to 10^{-4}. We consider two distinct metrics used to assess the tightness of relaxations. First, we consider the maximal pointwise distance from the convex relaxation to the concave relaxation, $\mathscr{P}(P) = \max_{p\in P}\left(f_{mlp}^{cc}(p) - f_{mlp}^{cv}(p)\right)$, which relates to the pointwise convergence properties of McCormick relaxations. Secondly, we consider a tightness metric for convex and concave relaxations, $\mathscr{H}(P) = \text{diam}\left(\left[\min_{p\in P} f_{mlp}^{cv}(p), \max_{p\in P} f_{mlp}^{cc}(p)\right]\right)$, which relates to the Hausdorff convergence properties of McCormick relaxations as discussed in [77]. We refer the reader to the original work for a full discussion of the underlying theory that motivates this metric.

The use of the envelopes improves the tightness of the derived relaxations of the MLP over naïve McCormick relaxations as illustrated in Figure 6. Under each distance metric, the relaxations of most activation functions exhibit quadratic convergence; a property required to avoid the clustering problem in spatial B&B [78, 79]. Further, as expected, the convex and concave envelopes of the activation functions result in a significant reduction in overestimation of the relaxations of the MLP. This improvement is most apparent under the $\mathscr{P}(P)$ metric. As previously hypothesized, despite the additional computational cost required to calculate the envelopes, it is expected that the reduction in overestimation will significantly speed up convergence of the B&B algorithm for deterministic global optimization of models containing ANNs.

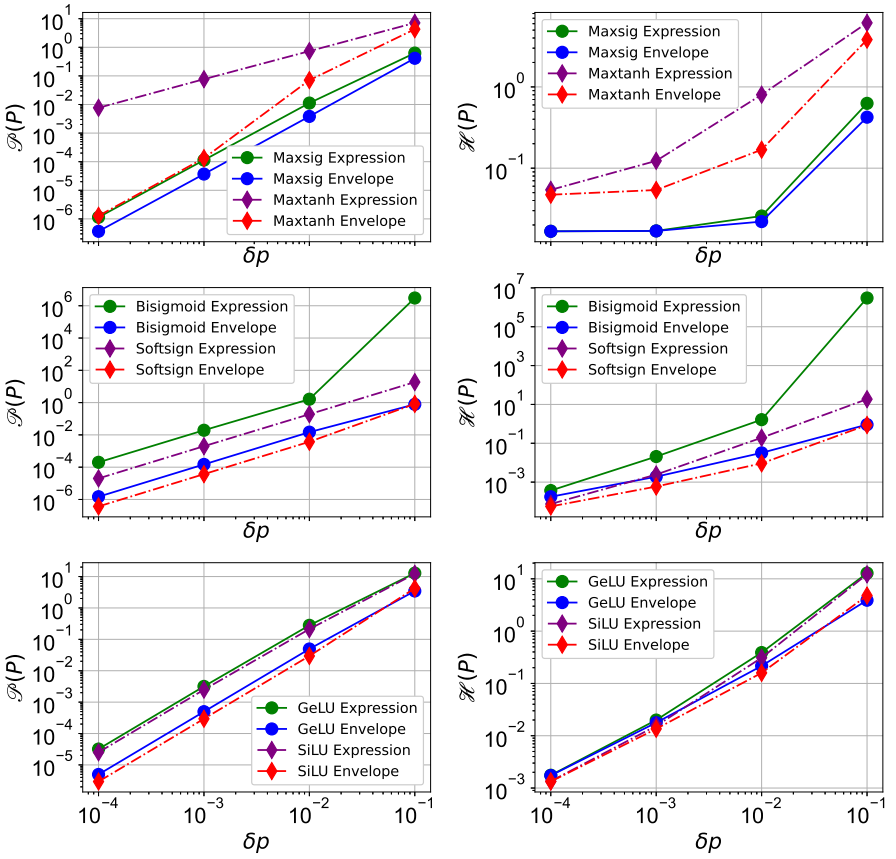

Fig. 6 A comparison of the tightness of the envelopes of activation functions is made against relaxations derived using the naïve McCormick relaxation approach. **Left:** pointwise convergence behavior is illustrated using the $\mathscr{P}(P)$ metric and **right:** the behavior under the relaxation tightness metric $\mathscr{H}(P)$, is shown. Under each metric, the relaxations generated using the envelopes of the activation functions outperform naïve McCormick relaxations. This comparison is made for (**top**) characteristic convex, (**middle**) convexoconcave, and (**bottom**) for the newer SiLU and GELU activation functions with relaxations from Theorem 1.

4 Global Optimization of ANNs

Optimization problems with ANN models [3] are formalized in this section. The vector of input variables is defined as $\mathbf{x} \in X \subseteq \mathbb{R}^{n_x}$, and the vector of output variables of an ANN is defined as $\mathbf{z} \in Z \subset \mathbb{R}^{n_z}$. $\mathbf{h} : X \times Z \to \mathbb{R}^{n_z}$ represents the general nonlinear network equations governed by an ANN model. $\mathbf{g} : X \times Z \to \mathbb{R}^{n_z}$ represents the inequalities that constrain the feasible region. $\phi : X \times Z \to \mathbb{R}$ represents the objective function. Generally, a *full-space* formulation denotes that the equality constraints governed by a system model are directly embedded. The formulation with ANNs embedded can be expressed as:

$$\begin{aligned} \min_{\mathbf{x}\in X, \mathbf{z}\in Z} \ & \phi(\mathbf{x}, \mathbf{z}) \\ \text{s.t.} \ & \mathbf{h}(\mathbf{x}, \mathbf{z}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}, \mathbf{z}) \leq \mathbf{0}. \end{aligned} \tag{12}$$

As introduced in Section 2.3, in a MLP, the network governing equations $\mathbf{h}(\mathbf{x}, \mathbf{z}) = \mathbf{0}$ can be calculated as an explicit input-output form: $\mathbf{z} = \mathbf{o}(\mathbf{x})$. Thus, the equality constraints can be eliminated and (12) can be reformulated to a *reduced-space* form:

$$\begin{aligned} \min_{\mathbf{x}\in X} \ & \phi(\mathbf{x}, \mathbf{o}(\mathbf{x})) \\ \text{s.t.} \ & \mathbf{g}(\mathbf{x}, \mathbf{o}(\mathbf{x})) \leq \mathbf{0}. \end{aligned} \tag{13}$$

Reduced-space methods with respect to deterministic global optimization originated in [80], where details were introduced for a method using a B&B algorithm with only a subset of the decision variables being branched on. This approach was further generalized for broader classes of model structures (e.g., [32, 81–83]). The core idea of a *reduced-space* method is to treat the vector of independent input variables $\mathbf{x}$ as the only decision variables of the optimization problem by eliminating the equality constraints and therefore the explicit dependence of the problem on auxiliary variables through intermediate computation and compositions.

In deterministic global optimization, a variation of the spatial B&B algorithm can be applied to solve nonconvex problems with formulations of (12) and (13) [52, 62, 63]. The B&B algorithm iteratively partitions the decision space into successively smaller subdomains and solves a sequence of upper-bounding and lower-bounding problems on each subdomain. Upper bounds are typically determined by solving nonconvex optimization problems in subdomains to either feasibility or local optimality. Lower bounds rely on convex and concave relaxations of the objective and constraints. As the algorithm proceeds, the best-found bounds are saved for comparison. By comparing the obtained upper and lower bounds, the algorithm converges to an ϵ-optimal global solution in finitely-many iterations, or a certification of infeasibility.

Many global optimizers, such as BARON [63, 84] and ANTIGONE [85], may introduce auxiliary variables to any of the formulations (12) when constructing subproblems, resulting in excessively large dimensionality subproblems with a high time cost due to the curse of dimensionality. This is especially true for ANN models as they include multiple layers, neurons, and network equations that inherently results in a large-scale optimization problem. In contrast, the EAGO [59] and MAiNGO [86] global solvers allow for the construction of relaxations directly from (13), which has been shown to reduce the computational complexity of solving subproblems and dramatically decreases solution time costs on several examples [3, 33, 80, 83, 87–89].

5 Numerical Experiments

We now illustrate how the use of envelopes described herein leads to a reduction in CPU run time and allows for a large variety of problems to be solved to deterministic global optimality. This is done by comparing the methods on a randomly generated benchmark library using the approach presented by Dolan and Moré [90]. The *performance* of a solver configuration s is taken to be the solution time $t_{p,s}$ in CPU seconds (single-threaded) for problem p. We consider the *performance ratio* on problem p by solver s to be the ratio of solver s performance to the best solver performance in the set:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}.$$

The *performance profile* of solver s on a particular benchmark set depicts the distribution function of the performance metric, $\rho_s(\tau)$; the probability that a performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio

$$\rho_s(\tau) = \frac{1}{n_p}\text{size}\{p \in \mathscr{P} : r_{p,s} \leq \tau\},$$

where $\mathscr{P}$ is the set of problems with $n_p = \text{card}(\mathscr{P})$. A plot comparing r_s for each solver configuration $s \in S$ will then illustrate the relative performance. For problems that terminate due to the specified time limit, the relative gap remaining can be compared to assess solver performance. The relative gap remaining is given by `(U - L)/ max(U, L)` where `U` is the upper bound (best feasible objective value) and `L` is the lower bound.

As the focus of this paper lies in the development of global optimization algorithms—and to the best of our knowledge, no benchmark library of ANN-embedded global optimization test problems exists—we choose to utilize a randomly generated library of 100 MLPs. Since the following analysis is motivated by a desire to compare the computational results between solvers and relaxation methods, we may conceptualize each random MLP as an ideally-trained MLP for some arbitrary function that serves to extricate the role of the optimization method from

Attribute	Values
Number of Input Variables	2 to 5
Number of Hidden Layers	1 to 4
Number of Neurons per Layer	2 to 5

Table 4 The range of values for each metaparameter used to generate the instances in the benchmark set are listed here.

the confounding effects of surrogate model structure and training methodology. Prior works have shown that the use of tanh envelopes in trained ANNs [3] and envelopes participating in trained Gaussian process models [6] lead to a decrease in overall solve times for optimization. As such, the use of randomly generated MLPs is expected to yield qualitatively similar results to trained models.

The MLPs used in the benchmark set were constructed by assigning weights and bias values from a uniform distribution randomly sampled within $[-1,1]$. The number of decision variables participating in the MLP, the number of layers, and the number of neurons per layer for each instance are randomly chosen from a uniform distribution within the ranges listed in Table 4. The objective function considered is a simple summation (14):

$$\phi(\mathbf{x},\mathbf{o}(\mathbf{x})) = \sum_i \mathbf{o}_i(\mathbf{x}). \tag{14}$$

The only constraints present in each problem are the box constraints on the decision variables, x_i, such that $x_i \in [-1,1]$ for $i = 1,\ldots,n$.

5.1 Implementation

All numerical experiments in this work were run on a single thread of an Intel Xeon E3-1270 v5 3.60/4.00GHz (base/turbo) processor with 16GB ECC RAM allocated to a virtual machine running the Ubuntu 18.04LTS operating system and Julia v1.6.2 [91]. Absolute and relative convergence tolerances for the B&B algorithm of 10^{-4} were specified for all example problems along with a maximum CPU time limit of 15 minutes (900 seconds). The EAGO.jl v0.7.0 package [59] was used to solve each optimization problem. Relaxations based on the envelopes of each activation function were implemented in the McCormick.jl [35] subpackage of EAGO.jl and are openly available. Validated interval arithmetic was computed using the package IntervalArithmetic.jl [92]. The Intel MKL package (2019 Update 2) [93] was used to perform all LAPACK [94, 95] and BLAS [96] routines. BARON v21.1.13 [63, 84] and SCIP 7.0.3[97] were used for performance comparisons. The data used with and generated from the following numerical examples are openly available in the Git repository established at https://github.com/PSORLab/RSActivationFunctions along with the corresponding problem formulations.

5.2 Benchmark Results

We now examine the impact of the envelopes on the solution times of a reduced-space global optimizer for solving deterministic global optimization problems with ANN models. The algebraic expressions for all activation functions considered in this study can be found in Tables 1 and 2, and Equations 6 and 7. Computational experiments were then conducted in which global optimization problems were solved for the benchmark suite of ANNs. To compare the performance of using the developed library of envelopes, the optimization problems were solved using both naïve McCormick relaxations (EAGO - McCormick) as well as the envelopes (EAGO - Envelope). These configurations are then compared to a state-of-the-art open-source solver, SCIP [97], and the state of the art commercial solver BARON [63, 84]. This comparison was made as SCIP, like EAGO, is open-source but implements the auxiliary variable method to constructing polyhedral relaxations used to compute lower bounds and refine the problem domain [97]. As SCIP does not support nonlinear objectives directly, a variable q was introduced and the problem was recast via the epigraph reformulation:

$$\begin{aligned} &\min_{\mathbf{x}\in X, q\in Q} \; q \\ &\text{s.t.} \; \sum_i \mathbf{o}_i(\mathbf{x}) \leq q. \end{aligned} \tag{15}$$

We note that EAGO performs this reformulation automatically as a preprocessing step. MLPs corresponding to sigmoid, softplus, silu, and gelu activation were considered. As illustrated by the performance profile plot depicted in Figure 7, EAGO generally outperforms SCIP on this limited benchmark set and the use of the activation function envelopes developed herein further improves computational performance, as evidenced by the the increased number of problems solved within the 15-minute limit detailed in Table 5. As shown in Tables 6 and 7, the use of envelopes in EAGO categorically increases the number of problems solved within 15 minutes for each activation function and further decreases the 15-minute average relative gap remaining for unsolved problems. Moreover, the shifted geometric mean solve times shown in Table 8 are reduced by using the envelopes presented herein.

While EAGO outperforms BARON with the new envelope functions for sigmoid and silu, we see significantly higher shifted geometric mean solve times for EAGO than BARON for softplus. While BARON is not an open-source solver, we may speculate on the potential reasons. We note that the median solve time for softplus problems is 0.2 CPU seconds, indicating that the geometric mean solve time is highly influenced by high solution time instances. The softplus activation function (also termed a “logistic loss" function) is known to have an epigraph that may be represented by the exponential cone and a disciplined convex programming approach used in BARON’s presolve phase indicates that softplus($\mathbf{a}^T\mathbf{x} + b$) is

itself convex [98, 99]. Taken together, these two factors indicate that BARON's presolve convexity detection may allow it to derive tighter bounds automatically. This highlights the potential that improved automatic convexity detection may have to further mitigate the overestimation of activation function relaxations as compared to the naïve McCormick approach.

We note that for a select number of poorly scaled problems, either BARON, EAGO, or SCIP may incorrectly terminate with a certificate of infeasibility. This may be attributed to the rounding errors encountered when computing polyhedral relaxations from convex and concave relaxations and their associated subgradients. EAGO currently makes use of a heuristic approach to ensure that only numerically-safe affine relaxations are added to subproblems. BARON v21.1.13 makes use of a state-of-the-art adaptive approach to resolve numerical difficulties. In either case, neither of these approaches are sufficient to fully resolve all numerical issues occurring in the benchmark set. Accordingly, additional work on resolving these numerical issues remains an active area of research. It should be noted that the presence of poorly scaled problems in the test set may be due to the use of randomly generated MLPs. Establishing a trained ANN benchmark set could be useful in providing some insight into this issue. However, in spite of this observation, we can see that the use of envelopes in this context leads to categorically improved computational performance relative to the naïve McCormick calculations.

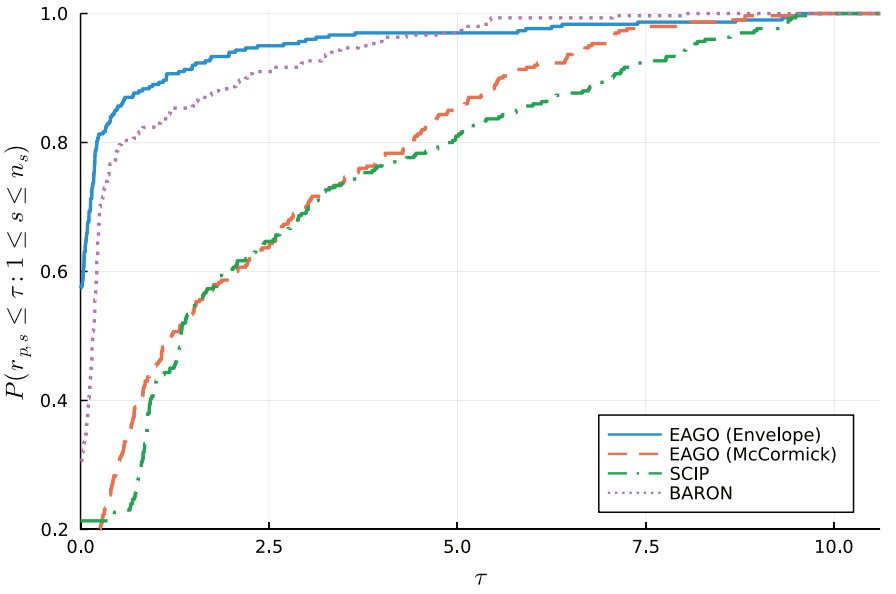

Fig. 7 As illustrated by the performance profiles shown for each solver configuration, computing relaxations using the envelopes leads to a substantial decrease in CPU solution time for a typical problem within the benchmark set when compared to either SCIP or the naïve McCormick approach implemented in EAGO. The EAGO (envelope) calculations (blue-solid) also outperform BARON for many activation function types, which is evident from the superior performance for $\tau < 5$.

Solver	Solved	Unsolved
EAGO (Envelope)	280 (93.3%)	20 (6.7%)
EAGO (Naïve McCormick)	260 (86.7%)	40 (13.3%)
SCIP	240 (80.0%)	60 (20.0%)
BARON	273 (91.0%)	27 (9.0%)

Table 5 The number of problems solved within 15 minutes by solver configuration in the benchmark set, are tabulated. Only sigmoid, softplus, and silu functions are used in these calculations for fair comparison since gelu is unsupported by both SCIP and BARON. EAGO and the developed envelopes outperform all other configurations in total problems solved.

Activation Function	EAGO (Envelope)	EAGO (McCormick)	SCIP	BARON
Sigmoid	96 (4)	94 (4)	91 (5)	94 (4)
Softplus	93 (7)	88 (7)	85 (8)	92 (7)
SiLU	91 (0)	78 (0)	64 (1)	87 (0)
GELU	76 (0)	59 (0)	N/A	N/A

Table 6 The number of benchmark problems solved within the 15-minute time limit are listed by solver configuration and activation function. The number of problems returning an infeasible result are given in parentheses for each condition. EAGO and the developed envelopes outperform all other configurations in total problems solved for each activation function.

Activation Function	EAGO (Envelope)	EAGO (McCormick)	SCIP	BARON
Sigmoid	N/A	2.4×10^{-3}	5.2×10^{1}	2.5×10^{-2}
Softplus	N/A	1.0×10^{-1}	7.3×10^{1}	8.9×10^{1}
SiLU	1.7×10^{-1}	9.9×10^{0}	2.1×10^{1}	4.2×10^{1}
GELU	9.5×10^{-1}	3.3×10^{1}	N/A	N/A

Table 7 The average relative gap remaining for any problems not solved within the 15-minute time limit are listed by solver configuration and activation function. For each activation funciton, EAGO with the developed envelopes have significantly smaller relative gaps at the 15-minute limit.

6 Conclusion

In this work, the convex/concave envelopes were developed for a variety of activation functions commonly used in ANN surrogate models. These included an identification of several activation functions that have not been previously implemented in existing libraries of relaxations and lack a general factorable form. Moreover, the McCormick arithmetic approach was shown to lead to weak relaxations for several common activation functions that have factorable representations. Particular attention was paid to the development of envelopes for the

Activation Function	EAGO (Envelope)	EAGO (McCormick)	SCIP	BARON
Sigmoid	1.09	4.16	6.37	1.54
Softplus	2.43	7.19	7.69	1.36
SiLU	5.04	36.77	43.11	9.58
GELU	19.62	82.69	N/A	N/A

Table 8 The shifted geometric mean of solve times $t_1, t_2, \ldots, t_n$ defined by $(\prod_{i=1}^{n}(t_i + s))^{1/n} - s$ are given by solver configuration and activation function with $s = 1$. EAGO using the envelopes developed herein outperforms naïve McCormick and SCIP on all activation functions examined. However, BARON outperforms all configurations examined for the Softplus function.

novel SiLU and GELU activation functions that do not belong in standard convexity classes (convex, convexoconcave, etc.) that have been previously addressed. Further, we demonstrated that the use of these envelopes provides desirable Hausdorff and pointwise convergence properties for the relaxations of the underlying activation functions.

Lastly, we generated benchmark results with respect to the performance of these relaxations when incorporated into a reduced-space global optimization routine using the EAGO optimizer. Using a randomly generated benchmark set of MLPs, we illustrated that the use of envelopes leads to a substantial reduction in run time and this reduced-space approach outperforms the full-space approach implemented in SCIP, leading to solving 13.3% more benchmark problems solved within the specified time limit. Moreover, this approach is comparable to the performance of the state-of-the-art commercial solver BARON. As such, the use of these envelopes provides a unilateral improvement when computing relaxations using a reduced-space optimization approach.

Two principal avenues for future work lie in the extension of these methods to continuous-time neural networks and deep ANNs with implicit representations. Continuous-time neural networks, such as continuous-time recurrent neural networks, may be incorporated into global optimization formulations using modern dynamic relaxation methods [42–45]. Deep ANNs with an implicit representation [46] may be addressed using fixed-point relaxation methods [33]. Lastly, particular attention should be paid to deep residual networks as the overestimation of affine relaxations computed via McCormick rules is minimal with particular respect to linear combinations. In any case, the further development of improved methods for constructing reduced-space relaxations of activation functions and standard layer structures that participate in ANNs, are expected to lead to improved computational performance and remain an intriguing area for future research.

Lastly, we note that future research and method development for the optimization of models containing trained ANNs could greatly benefit from the creation of a publicly available benchmark library containing relevant academic and industrial examples. To date, a dearth of trained ANNs has been made available from the corresponding modeling papers. As a result, numerical experiments for the

demonstration of performance of the established envelopes on trained systems has been inhibited. However, this work will be critical to further understanding the impacts of methods across applications.

Data Availability

The datasets generated and analysed during the current study are available in the Github repository, https://github.com/PSORLab/RSActivationFunctions.

Acknowledgements

Funding: This material is based upon work supported by the National Science Foundation under Grant No. 1932723. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Kahrs, O., Marquardt, W.: The validity domain of hybrid models and its application in process optimization. Chemical Engineering and Processing: Process Intensification **46**(11), 1054–1066 (2007). https://doi.org/10.1016/j.cep.2007.02.031

[2] Henao, C.A., Maravelias, C.T.: Surrogate-based superstructure optimization framework. AIChE Journal **57**(5), 1216–1232 (2010). https://doi.org/10.1002/aic.12341

[3] Schweidtmann, A.M., Mitsos, A.: Deterministic global optimization with artificial neural networks embedded. Journal of Optimization Theory and Applications **180**(3), 925–948 (2018). https://doi.org/10.1007/s10957-018-1396-0

[4] Williams, C., Rasmussen, C.: Gaussian processes for regression. Advances in Neural Information Processing systems **8**, 514–520 (1995). https://doi.org/10.5555/2998828.2998901

[5] Caballero, J.A., Grossmann, I.E.: An algorithm for the use of surrogate models in modular flowsheet optimization. AIChE Journal **54**(10), 2633–2650 (2008). https://doi.org/10.1002/aic.11579

[6] Schweidtmann, A.M., Bongartz, D., Grothe, D., Kerkenhoff, T., Lin, X., Najman, J., Mitsos, A.: Deterministic global optimization with gaussian processes embedded. Mathematical Programming Computation **13**(3), 553–581 (2021). https://doi.org/10.1007/s12532-021-00204-y

[7] Schweidtmann, A.M., Weber, J.M., Wende, C., Netze, L., Mitsos, A.: Obey validity limits of data-driven models through topological data analysis and one-class classification. Optimization and Engineering, 1–22 (2021). https://doi.org/10.1007/s11081-021-09608-0

[8] 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/

[9] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep

learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32 (NeurIPS 2019), pp. 8024–8035. Curran Associates, Inc., Vancouver, Canada (2019). https://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[10] Fahmi, I., Cremaschi, S.: Process synthesis of biodiesel production plant using artificial neural networks as the surrogate models. Computers & Chemical Engineering **46**, 105–123 (2012). https://doi.org/10.1016/j.compchemeng.2012.06.006

[11] Nagata, Y., Chu, K.H.: Optimization of a fermentation medium using neural networks and genetic algorithms. Biotechnology Letters **25**(21), 1837–1842 (2003). https://doi.org/10.1023/a:1026225526558

[12] Anna, H.R.S., Barreto, A.G., Tavares, F.W., de Souza, M.B.: Machine learning model and optimization of a PSA unit for methane-nitrogen separation. Computers & Chemical Engineering **104**, 377–391 (2017). https://doi.org/10.1016/j.compchemeng.2017.05.006

[13] Dornier, M., Decloux, M., Trystram, G., Lebert, A.M.: Interest of neural networks for the optimization of the crossflow filtration process. LWT - Food Science and Technology **28**(3), 300–309 (1995). https://doi.org/10.1016/s0023-6438(95)94364-1

[14] Nascimento, C.A.O., Giudici, R.: Neural network based approach for optimisation applied to an industrial nylon-6,6 polymerisation process. Computers & Chemical Engineering **22**, 595–600 (1998). https://doi.org/10.1016/s0098-1354(98)00105-7

[15] Hussain, M.A.: Review of the applications of neural networks in chemical process control — simulation and online implementation. Artificial Intelligence in Engineering **13**(1), 55–68 (1999). https://doi.org/10.1016/s0954-1810(98)00011-9

[16] Onel, M., Kieslich, C.A., Pistikopoulos, E.N.: A nonlinear support vector machine-based feature selection approach for fault detection and diagnosis: Application to the tennessee eastman process. AIChE Journal **65**(3), 992–1005 (2019). https://doi.org/10.1002/aic.16497

[17] Seong, Y., Park, C., Choi, J., Jang, I.: Surrogate model with a deep neural network to evaluate gas–liquid flow in a horizontal pipe. Energies **13**(4), 968 (2020). https://doi.org/10.3390/en13040968

[18] Villmann, T., Ravichandran, J., Villmann, A., Nebel, D., Kaden, M.: Investigation of activation functions for generalized learning vector quantization. International Workshop on Self-Organizing Maps, 179–188 (2019).

https://doi.org/10.1007/978-3-030-19642-4_18. Springer

[19] Xu, L., Chen, C.P.: Comparison and combination of activation functions in broad learning system. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 3537–3542 (2020). https://doi.org/10.1109/SMC42975.2020.9282871. IEEE

[20] Nader, A., Azar, D.: Searching for activation functions using a self-adaptive evolutionary algorithm. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, pp. 145–146 (2020). https://doi.org/10.1145/3377929.3389942

[21] Cristina, G.N.M., Sanchez, V.G.C., Villegas, O.O.V., Nandayapa, M., Dominguez, H.d.J.O., Azuela, J.H.S.: Study of the effect of combining activation functions in a convolutional neural network. IEEE Latin America Transactions **19**(5), 844–852 (2021). https://doi.org/10.1109/TLA.2021.9448319

[22] Fischetti, M., Jo, J.: Deep neural networks and mixed integer linear optimization. Constraints **23**(3), 296–309 (2018)

[23] Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., Vielma, J.P.: Strong mixed-integer programming formulations for trained neural networks. Mathematical Programming, 1–37 (2020)

[24] Kronqvist, J., Misener, R., Tsay, C.: Between steps: Intermediate relaxations between big-m and convex hull formulations. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research, pp. 299–314 (2021). Springer

[25] Tsay, C., Kronqvist, J., Thebelt, A., Misener, R.: Partition-based formulations for mixed-integer optimization of trained relu neural networks. Advances in Neural Information Processing Systems **34** (2021)

[26] Grimstad, B., Andersson, H.: ReLU networks as surrogate models in mixed-integer linear programs. Computers & Chemical Engineering **131**, 106580 (2019). https://doi.org/10.1016/j.compchemeng.2019.106580

[27] Schweidtmann, A.M., Huster, W.R., Lüthje, J.T., Mitsos, A.: Deterministic global process optimization: Accurate (single-species) properties via artificial neural networks. Computers & Chemical Engineering **121**, 67–74 (2019). https://doi.org/10.1016/j.compchemeng.2018.10.007

[28] Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. SIAM, Philadelpha, PA (2009). https://doi.org/10.1137/1.9780898717716

[29] Sahlodin, A.M., Chachuat, B.: Convex/concave relaxations of parametric odes

using taylor models. Computers & Chemical Engineering **35**(5), 844–857 (2011). https://doi.org/10.1016/j.compchemeng.2011.01.031

[30] McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: Part I — convex underestimating problems. Mathematical Programming **10**(1), 147–175 (1976). https://doi.org/10.1007/bf01580665

[31] Scott, J.K., Stuber, M.D., Barton, P.I.: Generalized McCormick relaxations. Journal of Global Optimization **51**(4), 569–606 (2011). https://doi.org/10.1007/s10898-011-9664-7

[32] Mitsos, A., Chachuat, B., Barton, P.I.: McCormick-based relaxations of algorithms. SIAM Journal on Optimization **20**(2), 573–601 (2009). https://doi.org/10.1137/080717341

[33] Stuber, M.D., Scott, J.K., Barton, P.I.: Convex and concave relaxations of implicit functions. Optimization Methods and Software **30**(3), 424–460 (2015). https://doi.org/10.1080/10556788.2014.924514

[34] Yajima, Y.: Convex envelopes in optimization problems: Convex envelopes in optimization problems. In: Floudas, C.A., Pardalos, P.M. (eds.) Encyclopedia of Optimization, pp. 343–344. Springer, Boston, MA (2001). https://doi.org/10.1007/0-306-48332-7_74

[35] Wilhelm, M.E., Gottlieb, R.X., Stuber, M.D.: PSORLab/McCormick.jl. Zenodo (2020). https://doi.org/10.5281/ZENODO.5749918. https://github.com/PSORLab/McCormick.jl

[36] Funahashi, K.-i., Nakamura, Y.: Approximation of dynamical systems by continuous time recurrent neural networks. Neural networks **6**(6), 801–806 (1993). https://doi.org/10.1016/S0893-6080(05)80125-X

[37] Haber, E., Ruthotto, L.: Stable architectures for deep neural networks. Inverse problems **34**(1), 014004 (2017). https://doi.org/10.1088/1361-6420/aa9a90

[38] Lu, Y., Zhong, A., Li, Q., Dong, B.: Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In: International Conference on Machine Learning, pp. 3276–3285 (2018). PMLR

[39] Ruthotto, L., Haber, E.: Deep neural networks motivated by partial differential equations. Journal of Mathematical Imaging and Vision **62**(3), 352–364 (2020). https://doi.org/10.1007/s10851-019-00903-1

[40] Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems, pp. 6572–6583 (2018). https://doi.org/10.5555/3327757.3327764

[41] Rackauckas, C., Innes, M., Ma, Y., Bettencourt, J., White, L., Dixit, V.: Diffeqflux.jl - A Julia library for neural differential equations. arXiv preprint arXiv:1902.02376 (2019) https://arxiv.org/abs/1902.02376

[42] Scott, J.K., Barton, P.I.: Improved relaxations for the parametric solutions of odes using differential inequalities. Journal of Global Optimization **57**(1), 143–176 (2013). https://doi.org/10.1007/s10898-012-9909-0

[43] Scott, J.K., Chachuat, B., Barton, P.I.: Nonlinear convex and concave relaxations for the solutions of parametric odes. Optimal Control Applications and Methods **34**(2), 145–163 (2013). https://doi.org/10.1002/oca.2014

[44] Wilhelm, M.E., Le, A.V., Stuber, M.D.: Global optimization of stiff dynamical systems. AIChE Journal (2019). https://doi.org/10.1002/aic.16836

[45] Song, Y., Khan, K.A.: Optimization-based convex relaxations for nonconvex parametric systems of ordinary differential equations. Mathematical Programming (2021). https://doi.org/10.1007/s10107-021-01654-x

[46] El Ghaoui, L., Gu, F., Travacca, B., Askari, A., Tsai, A.: Implicit deep learning. SIAM Journal on Mathematics of Data Science **3**(3), 930–958 (2021). https://doi.org/10.1137/20M1358517

[47] Celik, A.N., Kolhe, M.: Generalized feed-forward based method for wind energy prediction. Applied Energy **101**, 582–588 (2013). https://doi.org/10.1016/j.apenergy.2012.06.040

[48] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010). https://proceedings.mlr.press/v9/glorot10a.html

[49] Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1-3), 489–501 (2006). https://doi.org/10.1016/j.neucom.2005.12.126

[50] Medsker, L., Jain, L.C.: Recurrent Neural Networks: Design and Applications, pp. 64–67. CRC press, Boca Raton (1999). https://doi.org/10.1201/9781003040620

[51] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90

[52] Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches. Springer, Berlin, Germany (2013)

[53] Schweidtmann, A.M., Bongartz, D., Huster, W.R., Mitsos, A.: Deterministic global process optimization: Flash calculations via artificial neural networks. In: Computer Aided Chemical Engineering vol. 46, pp. 937–942. Elsevier, Amsterdam, Netherlands (2019). https://doi.org/10.1016/b978-0-12-818634-3.50157-0

[54] Chachuat, B.C.: MC++: Toolkit for Construction, Manipulation and Bounding of Factorable Functions (2020). https://omega-icl.github.io/mcpp/

[55] Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Communications of the ACM **60**(6), 84–90 (2017). https://doi.org/10.1145/3065386

[56] Lu, L., Shin, Y., Su, Y., Karniadakis, G.E.: Dying ReLU and initialization: Theory and numerical examples. Communications in Computational Physics **28**(5), 1671–1706 (2020). https://doi.org/10.4208/cicp.OA-2020-0165

[57] Clevert, D.-A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015) https://arxiv.org/abs/1511.07289

[58] Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Advances in Neural Information Processing Systems, pp. 971–980 (2017). https://doi.org/10.5555/3294771.3294864

[59] Wilhelm, M.E., Stuber, M.D.: EAGO.jl: easy advanced global optimization in Julia. Optimization Methods and Software, 1–26 (2020). https://doi.org/10.1080/10556788.2020.1786566

[60] Bompadre, A., Mitsos, A.: Convergence rate of McCormick relaxations. Journal of Global Optimization **52**(1), 1–28 (2011). https://doi.org/10.1007/s10898-011-9685-2

[61] Kannan, R., Barton, P.I.: The cluster problem in constrained global optimization. Journal of Global Optimization **69**(3), 629–676 (2017). https://doi.org/10.1007/s10898-017-0531-z

[62] Ryoo, H.S., Sahinidis, N.V.: A branch-and-reduce approach to global optimization. Journal of Global Optimization **8**(2), 107–138 (1996). https://doi.org/10.1007/bf00138689

[63] Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Mathematical Programming **103**(2), 225–249 (2005). https://doi.org/10.1007/s10107-005-0581-8

[64] Nair, V., Hinton, G.E.: Rectified linear units improve restricted

boltzmann machines. ICML: Proceedings of the 27th International Conference on Machine Learning, 807–814 (2010). https://doi.org/10.5555/3104322.3104425

[65] He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1026–1034. IEEE, Santiago, Chile (2015). https://doi.org/10.1109/iccv.2015.123

[66] Eger, S., Youssef, P., Gurevych, I.: Is it time to swish? Comparing deep learning activation functions across NLP tasks. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (2018). https://doi.org/10.18653/v1/d18-1472

[67] Zheng, H., Yang, Z., Liu, W., Liang, J., Li, Y.: Improving deep neural networks using softplus units. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–4 (2015). https://doi.org/10.1109/IJCNN.2015.7280459. IEEE

[68] Nwankpa, C.E., Ijomah, W., Gachagan, A., Marshall, S.: Activation functions: comparison of trends in practice and research for deep learning. In: 2nd International Conference on Computational Sciences and Technology, pp. 124–133 (2021)

[69] Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2**(5), 359–366 (1989). https://doi.org/10.1016/0893-6080(89)90020-8

[70] Elliott, D.L.: A better activation function for artificial neural networks. Technical report, Institute for Systems Research (1993). http://hdl.handle.net/1903/5355

[71] Sahlodin, A.M.: Global optimization of dynamic process systems using complete search methods. PhD thesis, McMaster University (2013). https://macsphere.mcmaster.ca/handle/11375/12803

[72] Hendrycks, D., Gimpel, K.: Gaussian error linear units (GELUs). arXiv preprint (2016) https://arxiv.org/abs/1606.08415

[73] Elfwing, S., Uchibe, E., Doya, K.: Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. Neural Networks **107**, 3–11 (2018). https://doi.org/10.1016/j.neunet.2017.12.012

[74] Elfwing, S., Uchibe, E., Doya, K.: Expected energy-based restricted boltzmann machine for classification. Neural Networks **64**, 29–38 (2015). https://doi.org/10.1016/j.neunet.2014.09.006

[75] Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions. arXiv preprint (2017) https://arxiv.org/abs/1710.05941

[76] Chen, J., Revels, J.: Robust benchmarking in noisy environments. arXiv e-prints (2016) arXiv:1608.04295 [cs.PF]

[77] Najman, J., Mitsos, A.: Convergence analysis of multivariate McCormick relaxations. Journal of Global Optimization **66**(4), 597–628 (2016). https://doi.org/10.1007/s10898-016-0408-6

[78] Du, K., Kearfott, R.B.: The cluster problem in multivariate global optimization. Journal of Global Optimization **5(3)**(3), 253–265 (1994). https://doi.org/10.1007/bf01096455

[79] Wechsung, A., Schaber, S.D., Barton, P.I.: The cluster problem revisited. Journal of Global Optimization **58**(3), 429–438 (2014). https://doi.org/10.1007/s10898-013-0059-9

[80] Epperly, T.G.W., Pistikopoulos, E.N.: A reduced space branch and bound algorithm for global optimization. Journal of Global Optimization **11**(3), 287–311 (1997). https://doi.org/10.1023/A:1008212418949

[81] Stuber, M.D.: Evaluation of process systems operating envelopes. PhD thesis, Massachusetts Institute of Technology (2012). https://doi.org/10.13140/2.1.1775.4409

[82] Wechsung, A.: Global optimization in reduced space. PhD thesis, Massachusetts Institute of Technology (2014). https://dspace.mit.edu/handle/1721.1/87131

[83] Bongartz, D., Mitsos, A.: Deterministic global optimization of process flowsheets in a reduced space using McCormick relaxations. Journal of Global Optimization **69**(4), 761–796 (2017). https://doi.org/10.1007/s10898-017-0547-4

[84] Sahinidis, N.V.: BARON 21.1.13: Global Optimization of Mixed-Integer Nonlinear Programs, *User's Manual*. (2017). https://www.minlp.com/downloads/docs/baron%20manual.pdf

[85] Misener, R., Floudas, C.A.: ANTIGONE: Algorithms for continuous/integer global optimization of nonlinear equations. Journal of Global Optimization **59**(2-3), 503–526 (2014). https://doi.org/10.1007/s10898-014-0166-2

[86] Bongartz, D., Najman, J., Sass, S., Mitsos, A.: MAiNGO: McCormick based algorithm for mixed integer nonlinear global optimization. Process Systems Engineering (AVT. SVT), RWTH Aachen University (2018). https://git.rwth-aachen.de/avt-svt/public/maingo

[87] Kearfott, R.B., Castille, J., Tyagi, G.: A general framework for convexity analysis in deterministic global optimization. Journal of Global Optimization **56**(3), 765–785 (2013). https://doi.org/10.1007/s10898-012-9905-4

[88] Khan, K.A., Watson, H.A.J., Barton, P.I.: Differentiable McCormick relaxations. Journal of Global Optimization **67**(4), 687–729 (2016). https://doi.org/10.1007/s10898-016-0440-6

[89] Khan, K.A., Wilhelm, M., Stuber, M.D., Cao, H., Watson, H.A.J., Barton, P.I.: Corrections to: Differentiable McCormick relaxations. Journal of Global Optimization **70**(3), 705–706 (2018). https://doi.org/10.1007/s10898-017-0601-2

[90] Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical programming **91**(2), 201–213 (2002). https://doi.org/10.1007/s101070100263

[91] Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM Review **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671

[92] Sanders, D.P., Benet, L., lucaferranti, Agarwal, K., Richard, B., Grawitter, J., Gupta, E., Herbst, M.F., Forets, M., yashrajgupta, Hanson, E., van Dyk, B., Rackauckas, C., Vasani, R., Micluța-Câmpeanu, S., Olver, S., Koolen, T., Wormell, C., Vázquez, F.A., TagBot, J., O'Bryant, K., Carlsson, K., Piibeleht, M., Reno, Deits, R., Holy, T., Kaluba, M., matsueushi: JuliaIntervals/IntervalArithmetic.jl: V0.18.2. https://doi.org/10.5281/zenodo.4739394

[93] Fedorov, G., Nguyen, K.T., Harrison, P., Singh, A.: Intel Math Kernel Library 2019 Update 2 Release Notes. (2019). https://software.intel.com/en-us/mkl

[94] Anderson, E., Bai, Z., Bischof, C., Blackford, L.S., Demmel, J., Dongarra, J., Croz, J.D., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA (1999). https://doi.org/10.1137/1.9780898719604

[95] Wang, E., Zhang, Q., Shen, B., Zhang, G., Lu, X., Wu, Q., Wang, Y.: Intel math kernel library. In: High-Performance Computing on the Intel® Xeon Phi™, pp. 167–188. Springer, New York, NY (2014). https://doi.org/10.1007/978-3-319-06486-4_7

[96] Blackford, L.S., Petitet, A., Pozo, R., Remington, K., Whaley, R.C., Demmel, J., Dongarra, J., Duff, I., Hammarling, S., Henry, G., Heroux, M.: An updated set of basic linear algebra subprograms (BLAS). ACM Transactions on Mathematical Software **28**(2), 135–151 (2002). https://doi.org/10.1145/567806.567807

[97] Vigerske, S., Gleixner, A.: SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework.

Optimization Methods and Software **33**(3), 563–593 (2018). https://doi.org/10.1080/10556788.2017.1335312

[98] Grant, M., Boyd, S., Ye, Y.: In: Liberti, L., Maculan, N. (eds.) Disciplined convex programming, pp. 155–210. Springer, Boston, MA (2006). https://doi.org/10.1007/0-387-30528-9_7

[99] Khajavirad, A., Sahinidis, N.V.: A hybrid LP/NLP paradigm for global optimization relaxations. Mathematical Programming Computation **10**(3), 383–421 (2018). https://doi.org/10.1007/s12532-018-0138-5