

THREE DECADES OF ACTIVATIONS: A COMPREHENSIVE SURVEY OF 400 ACTIVATION FUNCTIONS FOR NEURAL NETWORKS

A PREPRINT

 **Vladimír Kunc**

Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague
kuncvlad@fel.cvut.cz

 **Jiří Kléma**

Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague
klema@fel.cvut.cz

February 15, 2024

ABSTRACT

Neural networks have proven to be a highly effective tool for solving complex problems in many areas of life. Recently, their importance and practical usability have further been reinforced with the advent of deep learning. One of the important conditions for the success of neural networks is the choice of an appropriate activation function introducing non-linearity into the model. Many types of these functions have been proposed in the literature in the past, but there is no single comprehensive source containing their exhaustive overview. The absence of this overview, even in our experience, leads to redundancy and the unintentional rediscovery of already existing activation functions. To bridge this gap, our paper presents an extensive survey involving 400 activation functions, which is several times larger in scale than previous surveys. Our comprehensive compilation also references these surveys; however, its main goal is to provide the most comprehensive overview and systematization of previously published activation functions with links to their original sources. The secondary aim is to update the current understanding of this family of functions.

Keywords adaptive activation functions, deep learning, neural networks

1 Introduction

Neural networks — and deep learning in particular — have exhibited remarkable success in addressing diverse challenges across various fields. They stand as state-of-the-art approaches, showcasing their prowess in solving complex and intricate problems. At the heart of these networks, activation functions (AFs) play an important role by introducing nonlinearity to neural network layers. In the absence of nonlinear AFs, typical neural networks would only model a weighted sum of inputs, limiting their capacity to capture intricate relationships within the data.

The choice of activation functions profoundly influences a network’s ability to learn and generalize, directly impacting its performance across a spectrum of tasks. Effective activation functions possess several key properties, as outlined by Dubey, Singh, and Chaudhuri in [1]: a) introducing non-linear curvature to enhance training convergence within the optimization landscape; b) maintaining an unobstructed gradient flow during training; c) ensuring a minimal increase in the computational complexity of the model; and d) preserving the distribution of data to optimize the network’s training.

There are many activation functions proposed in the literature in the last three decades — some more computationally complex or with higher performance than others. However, further research of the activation functions is hampered by the absence of a consolidated list. This gap leads to the inadvertent reinvention of existing activation functions and the independent proposal of identical or very similar ones, resulting in a wasteful consumption of research resources. Even comprehensive surveys and reviews, such as those by Dubey, Singh, and Chaudhuri [1] and Apicella et al. [2], often omit numerous activation functions present in the literature; furthermore, these reviews are also a bit older and many new activation functions emerged since then. This oversight can lead to instances where an AF is redundantly proposed as novel, despite its prior introduction in the literature — e.g., rectified power unit (RePU) (section 3.6.39), dual

parametric ReLU (DPRReLU) (section 4.2.20), truncated rectified (TRec) (section 3.6.21), ReLU-Swish (section 3.6.46), and bounded ReLU (BReLU) (section 3.6.16). By providing a more extensive list of available activation functions, we aim to avoid such redundancy and promote faster advances in the research of activation functions in neural networks.

To address this issue, we strive to provide an extensive and consolidated list of available AFs. This survey aims to prevent redundancy, eliminate the reinvention of established AFs to promote innovation, and accelerate the advancement of research in the field of neural networks. By offering a comprehensive resource, we aim to promote efficiency and innovation in the exploration of AFs within the field.

It is important to note that our contribution primarily focuses on providing a comprehensive list of AFs rather than conducting extensive benchmarks or in-depth analyses. The breadth of the compilation encompasses a wide array of AFs, making a detailed benchmark or a deeper analysis beyond the scope of this work. Our aim is to provide researchers with a foundational resource that facilitates informed decision-making in selecting AFs for neural networks, recognizing that a more exhaustive exploration or detailed analysis would necessitate a dedicated and focused effort beyond the scope of this comprehensive listing. The presented overview is limited to real-valued activation functions; complex-valued neural networks (e.g., [3–16], brief overview available in [17, 18]), bicomplex-valued neural networks (e.g., [19]), quaternion-valued neural networks (e.g., [20–24]), photonic neural networks (e.g., [25]), fuzzy neural networks (e.g., [26–31]), AFs for probabilistic boolean logic (e.g., [32]), quantum AFs (e.g., [33]) and others are out of the scope of this work.¹

We have chosen to categorize AFs into two main classes: fixed AFs (section 3) and adaptive activation functions (AAFs) (section 4), the latter having a parameter that is trained alongside the other weights in a network. Although instances exist where AFs are virtually identical, differing only in the presence of a particular adaptive parameter (e.g., swish (see section 4.4.1) and SiLU (see section 3.3)), this classification proves valuable. AAFs, by virtue of their parameterization, offer an added layer of flexibility in capturing complex relationships within the data during the training process.

2 Literature review

There are several reviews of AFs available in the literature; however, most of them encompass only the most commonly known AFs. While this is sufficient for an overview for newcomers to the field, it does not allow for efficient research of AFs themselves. Probably the most extensive review is the [1] from 2022, which lists over 70 AFs and provides a benchmark for 18 of them. Other reviews works containing list of AFs include [2, 18, 34–46].

While there are several existing works [1, 18, 37, 42, 47–99] that offer benchmarks and empirical comparisons of various AFs, it is unfortunate that these studies are often constrained by a limited selection of AFs. Typically, the focus is centered on the most well-known and widely used functions, neglecting the broader spectrum of AFs available in the literature.

To avoid the manual selection of AFs, many researchers resort to various optimization approaches to find the optimal AF for their problems. e.g. an evolutionary approach was used to evolve the optimal activation function in [35, 100–116] and grid search using artificial data was used in [117]. Another search for the optimal activation functions was presented in [49] where several simple activation functions were found to perform remarkably well. These automatic approaches might be used for evolving the activation functions (e.g., [100, 105]) or for selecting the optimal activation function for a given neuron (e.g., [108, 118]). While evolved activation function may perform well for a given problem, they also might be very complex — e.g., evolved activation functions in [105]. The complexity of an activation function is also important characteristic as it significantly influences the computational efficiency of a neural network; however, this might be mitigated by efficient implementations (including hardware implementations) of such activation functions (e.g., [119–129]). An empirical analysis of computational efficiency and power consumption of various AFs is available in [130].

3 Classical activation functions

First, our discussion delves into fixed activation functions, devoid of adaptive parameters. This category of activation functions represents the basic type that was predominantly employed in the initial neural network architectures and continues to be prevalent today. Fixed activation functions, such as the logistic sigmoid and hyperbolic tangent, are characterized by their predetermined mathematical formulations, where the activation output solely depends on the input without the introduction of any trainable parameters.

¹While these kinds of neural networks (NNs) are not discussed throughout this work, some of these approaches will use AFs presented in this work.

3.1 Binary activation function

The binary activation function (binary AF) — also called a step function — is a simple yet important activation function used in neural networks [131]. It assigns an output value of 1 if the input is positive or zero and an output value of 0 if the input is negative [2]. Mathematically, it can be defined as follows:

$$f(z) = \begin{cases} 1, & z \geq 0, \\ 0, & z < 0. \end{cases} \quad (1)$$

Similar to binary activation function is the sign function, which produces an output value of -1 if the input is negative and 1 if it is positive (and 0 for outputs that are exactly zero) [2]. Since the sign and the binary activation functions have nearly exact properties from the point of view of neural networks, only the binary activation function is mentioned, but the points hold similarly for the sign activation function.

The main advantage of the binary activation function is that it is straightforward and computationally efficient to implement. It does not involve complex mathematical operations, making it suitable for networks with low computational resources or for hardware implementations [132, 133]. However, the binary activation function has one glaring disadvantage - the lack of differentiability. The binary activation function is not differentiable at the point of discontinuity ($x = 0$) and is zero elsewhere. This poses challenges for optimization algorithms that rely on gradients, such as backpropagation (BP), since the gradient is noninformative [48, 131, 134]. Since the gradient-based methods are used predominantly, the binary activation function is used very rarely and is important mainly for historical reasons as it was used in the original perceptron [48, 135].

3.2 Sigmoid family of activation functions

Various smoothed variants of the binary activation functions (sigmoids) are commonly used; the most common is the logistic function — the standard logistic sigmoid function was dominant in the field prior the introduction of rectified linear unit (ReLU) (see section 3.6) [136], the logistic function is often called just sigmoid in the literature which is also used throughout this work for brevity (unless specified otherwise, *sigmoid* is equivalent to standard logistic function in the text). Standard logistic function is defined as

$$f(z) = \sigma(z) = \frac{1}{1 + \exp(-z)}. \quad (2)$$

The logistic sigmoid was a popular choice since its output values can be interpreted as the probability that a binary variable is 1 [136] since it squashes the input to the interval $(0, 1)$ [1]. The problem of sigmoid activation functions is that they saturate — they saturate when their input z is either a large positive number or a large negative number, which makes gradient-based learning difficult [1, 136]; therefore their use in feedforward networks is usually discouraged [136]. Another option, albeit significantly less popular in artificial neural networks (ANNs), is the probit AF [137], which is just the cumulative standard normal distribution function used as an AF [137].

Another popular sigmoid function is the hyperbolic tangent (tanh) activation function which is just scaled and shifted logistic sigmoid

$$\tanh(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)} = 2\sigma(2z) - 1. \quad (3)$$

Similarly as the logistic sigmoid, the tanh also squashes the inputs; however, it squashes them to the interval $(-1, 1)$. The tanh function is often advantageous over the logistic sigmoid function because it is centered around zero and it is similar to the identity function near zero, which makes training of a network easier if the activations are kept small [136]. Nevertheless, the tanh function saturates similarly as does the logistic sigmoid and therefore similarly suffers from the vanishing gradients [1]. Computationally efficient approximation of the tanh activation functions based on splines were proposed in [138] — *tanh36* based on approximation relying on 36 equidistant points and *tanh3* using only 3 points. Scaled variant $\tanh\left(\frac{z}{2}\right)$ was used in [139]. The linearized unit (LRTanh) is a tanh variant used together with modified BP that substitutes a different activation function derivative proposed in [140]. There are also approximations of the logistic sigmoid and tanh that are meant to speed up the computations; e.g., pRPPSG [141] and other similar piecewise approximations [142, 143].

A scaled version of the logistic sigmoid function was proposed in [144] with the motivation to have the same linear regimes as the tanh and relu activation functions when initialized with the popular normalized initialized method proposed in [145]. The scaled version used fixed parameters

$$f(z) = 4\sigma(z) - 2. \quad (4)$$

A more complicated variant named n-sigmoid was proposed in [146]; however, it seems that the formula presented in the paper is not as the authors intended and, therefore, we omit this AF from the list.

3.2.1 Shifted and scaled sigmoid (SSS)

The shifted and scaled sigmoid (SSS) was used in [147]; it is the logistic sigmoid with horizontal scaling and translation defined as

$$f(z) = \sigma(a(z - b)) = \frac{1}{1 + \exp(-a(z - b))}, \quad (5)$$

where a and b are predetermined parameters; Arai and Imamura used $a = 0.02$ and $b = 600$.

3.2.2 Variant sigmoid function (VSF)

The variant sigmoid function (VSF) is an older parametric variant of the logistic sigmoid proposed in [148]. It is defined as

$$f(z) = a\sigma(bz) - c = \frac{a}{1 + \exp(-bz)} - c, \quad (6)$$

where a , b , and c are predetermined parameters [148].

3.2.3 Scaled hyperbolic tangent

A parametric version called scaled hyperbolic tangent (stanh) was used in [149]:

$$f(z) = a \tanh(b \cdot z), \quad (7)$$

where a and b are fixed hyperparameters that control the scaling of the function. Lecun et al. proposed using $a = 1.7159$ and $b = \frac{2}{3}$

A similar concept was analyzed in [150] where sigmoids with bi-modal derivatives were used as activation functions. An example of such a function is

$$f(z) = \frac{1}{2} \left(\frac{1}{1 + \exp(-z)} + \frac{1}{1 + \exp(-z - b)} \right), \quad (8)$$

where b is a hyperparameter [150]; similarly, additional three activation functions with bi-modal derivatives were proposed in [150].

3.2.4 Arctan

The arctangent (arctan) function and its variation were used as activation functions in [151]:

$$f(z) = \tan^{-1}(z). \quad (9)$$

The arctan resembles a logistic sigmoid activation, however, it covers wider range $(-\frac{\pi}{2}, \frac{\pi}{2})$ [151]. The arctan and several its variation were compared with the tanh, ReLU, leaky ReLU (LReLU), logistic sigmoid activation, and swish in [151]; the best-performing functions in the presented experiments were the arctan and its variation arctanGR [151]. Interestingly, the arctan was used as an AF twenty years earlier in [152]. The arctanGR is a scaled version of the arctan and is defined as

$$f(z) = \frac{\tan^{-1}(z)}{\frac{1+\sqrt{2}}{2}}. \quad (10)$$

Other scaling variants such as division by the π , $\frac{1+\sqrt{5}}{2}$, or the Euler number are presented in [153].

3.2.5 Sigmoid-Algebraic activation function

The Sigmoid-Algebraic is a sigmoid variant defined in [154]. It is defined as

$$f(z) = \frac{1}{1 + \exp\left(-\frac{z(1+a|z|)}{1+|z|(1+a|z|)}\right)}, \quad (11)$$

where $a \geq 0$ is a parameter [154].

3.2.6 Triple-state sigmoid

The triple-state sigmoid unit (TS-sigmoid) is a cascaded AF similar to TS-swish (see section 3.3.6) [154]; it is defined as

$$f(z) = \frac{1}{1 + \exp(-z)} \left(\frac{1}{1 + \exp(-z)} + \frac{1}{1 + \exp(-z + a)} + \frac{1}{1 + \exp(-z + b)} \right), \quad (12)$$

where a and b are fixed parameters [154].

3.2.7 Improved logistic sigmoid

The improved logistic sigmoid is yet another sigmoid based activation function designed to deal with the vanishing gradient problem

$$f(z) = \begin{cases} a(z - b) + \sigma(b), & z \geq b, \\ \sigma(z), & -b < z < b, \\ a(z + b) + \sigma(b), & z \leq -b, \end{cases} \quad (13)$$

where a and b are fixed parameters [155]; a controls the slope and b is a thresholding parameter. The authors recommend a bound on the slope parameter a :

$$a > a_{\min} = \frac{\exp(-b)}{(1 + \exp(-b))^2}. \quad (14)$$

Even though the parameters are fixed during the training of a network, a procedure for presetting them based on the network and data was proposed in [155]. The output range of the sigmoid-weighted linear unit (SiLU) is $(-\infty, \infty)$ [1]. The authors Qin, Wang, and Zou also showed that the improved logistic sigmoid AF has a higher convergence speed than the logistic sigmoid AF [155].

3.2.8 Combination of the sigmoid and linear activation (SigLin)

A SigLin² was used as an AF in [156]. The SigLin is defined as

$$f(z) = \sigma(z) + az, \quad (15)$$

where $\sigma(z)$ is the logistic sigmoid AF and a is a fixed parameter [156]; however, this AF was used only in a modified optimization procedure [156]. Roodschild, Gotay Sardiñas, and Will experimented with $a \in \{0, 0.05, 0.1, 0.15\}$ [156].

3.2.9 Penalized hyperbolic tangent

A penalized hyperbolic tangent (ptanh) the LReLU (see section 3.6) but uses the tanh function instead of the linear function [144]:

$$f(z) = \begin{cases} \tanh(z), & z \geq 0, \\ \frac{\tanh(z)}{a}, & z < 0, \end{cases} \quad (16)$$

where $a \in (1, \infty)$. This function has similar values near 0 as the LReLU with identical parameter a as they both share the same Taylor expansion up to the first order [144]; however this function saturates to $-\frac{1}{a}$ for $z \rightarrow -\infty$ and to 1 for $z \rightarrow \infty$ [144]. The ptanh AF was found to perform consistently well for various natural language processing (NLP) tasks compared to ReLU, LReLU and several other activation functions [50].

3.2.10 Soft-root-sign (SRS)

A soft-root-sign (SRS) activation function is a parametric, smooth, non-monotonic, and bounded activation function [157]. It is defined as

$$f(z) = \frac{z}{\frac{z}{a} + \exp(-\frac{z}{b})}, \quad (17)$$

where a and b are predetermined parameters [157]; the authors Li and Zhou propose using $a = 2$ and $b = 3$ whereas the parameters are said to be learnable in [1]. The output range of SRS is $\left[\frac{ab}{b-ae}, a\right]$ [1, 157]. The performance of the SRS was demonstrated using the CIFAR-10 and CIFAR-100 [158] task in comparison with the ReLU (see section 3.6 for the description of the ReLU family of AFs), LReLU, parametric rectified linear unit (PReLU), softplus, exponential linear unit (ELU), scaled ELU (SELU), and swish [157].

²This abbreviation is used only in this work; Roodschild, Gotay Sardiñas, and Will did not name the function in [156].

3.2.11 Soft clipping (SC)

The soft clipping (SC) [159, 160] AF is another bounded AF; it is approximately piecewise linear in the range $z \in (0, 1)$ and it is defined as

$$f(z) = \frac{1}{a} \ln \left(\frac{1 + \exp(az)}{1 + \exp(a(z-1))} \right), \quad (18)$$

where a is a fixed parameter [160].

3.2.12 Hexpo

The Hexpo activation function [161] was proposed in order to minimize the problem of vanishing gradient [1]; it resembles a tanh activation function with scaled gradients [1]:

$$f(z) = \begin{cases} -a \left(\exp\left(-\frac{z}{b}\right) - 1 \right), & z \geq 0, \\ c \left(\exp\left(-\frac{z}{d}\right) - 1 \right), & z < 0, \end{cases} \quad (19)$$

where a , b , c , and d are fixed parameters. While the parameters could be trainable in theory, it is not recommended as it would lead to the vanishing gradient problem [161]. The Hexpo functions allow for control over the gradient by tuning the parameters a , b , c , and d and the ratios $\frac{a}{b}$ and $\frac{c}{d}$ — with increasing the ratios $\frac{a}{b}$ or $\frac{c}{d}$, the rate of gradient decay to zero decreases; increasing only a and c scales the gradient around the origin up [161].

3.2.13 Softsign

A softsign activation function is a smooth activation function similar to the tanh activation; however, it is less prone to vanishing gradients [47]. It is defined as

$$f(z) = \frac{z}{1 + |z|}, \quad (20)$$

where $|z|$ denotes the absolute value of z [47].

3.2.14 Smooth step

The smooth step is a sigmoid AF; it is defined as

$$f(z) = \begin{cases} 1 & z \geq \frac{a}{2}, \\ -\frac{2}{a^3}z^3 + \frac{3}{2a}z + \frac{1}{2}, & -\frac{a}{2} \leq z \leq \frac{a}{2}, \\ 0 & z \leq -\frac{a}{2}, \end{cases} \quad (21)$$

where a is a fixed hyperparameter [162].

3.2.15 Elliott activation function

Elliott activation function is one of the earliest proposed activation functions to replace to replace the logistic sigmoid or tanh activation functions [163]; the Elliott AF is a scaled and translated softsign AF. It is defined as [1, 34]

$$f(z) = \frac{0.5z}{1 + |z|} + 0.5. \quad (22)$$

The output of the Elliott activation functions is in range $[0, 1]$ [1, 34]. The main advantage of the Elliott AF is that it can be calculated much faster than the logistic sigmoid [164].

3.2.16 Sinc-Sigmoid

The Sinc-Sigmoid is a sigmoid-based AF proposed in [154]. It is defined as

$$f(z) = \text{sinc}(\sigma(z)), \quad (23)$$

where $\text{sinc}(x)$ is the unnormalized³ sinc function [154].

³Koçak and Üstündağ Şiray did not specify whether it is the normalized or unnormalized variant. Still, they provided the derivative of the Sinc-Sigmoid, which suggests that the unnormalized variant was used.

3.2.17 Sigmoid-Gumbel activation function

The Sigmoid-Gumbel (SG) is a non-adaptive AF proposed recently in [165]; it is defined as

$$f(z) = \frac{1}{1 + \exp(-z)} \exp(-\exp(-z)). \quad (24)$$

3.2.18 NewSigmoid

The NewSigmoid is a sigmoid variant proposed in [166]. It is defined as

$$f(z) = \frac{\exp(z) - \exp(-z)}{\sqrt{2}(\exp(2z) + \exp(-2z))}. \quad (25)$$

3.2.19 Root2sigmoid

The root2sigmoid is another sigmoid variant proposed in [166]. It is defined⁴ as

$$f(z) = \frac{\sqrt{2}^z - \sqrt{2}^{-z}}{2\sqrt{2}\sqrt{2\left(\sqrt{2}^{2z} + \sqrt{2}^{-2z}\right)}}. \quad (26)$$

3.2.20 LogLog

The LogLog is a simple AF proposed in [137]; it is defined as

$$f(z) = \exp(-\exp(-z)). \quad (27)$$

The LogLog, cLogLog (see section 3.2.21) were used in NNs for forecasting financial time-series in [137].

3.2.21 Complementary Log-Log (cLogLog)

The complementary LogLog (cLogLog) is another simple AF proposed in [137] complementing the LogLog (see section 3.2.20); it is defined as

$$f(z) = 1 - \exp(-\exp(-z)). \quad (28)$$

The variant called modified cLogLog (cLogLogm) [137] was also proposed:

$$f(z) = 1 - 2 \exp(-0.7 \exp(-z)). \quad (29)$$

3.2.22 SechSig

The SechSig [167] is another AF utilizing the logistic sigmoid in its definition; it is defined as

$$f(z) = (z + \operatorname{sech}(z)) \sigma(z). \quad (30)$$

Közkurt et al. also proposed a parametric version which we will call parametric SechSig (pSechSig):

$$f(z) = (z + a \cdot \operatorname{sech}(z + a)) \sigma(z), \quad (31)$$

where a is a fixed parameter [167].

3.2.23 TanhSig

The TanhSig [167] is an AF similar to SechSig; it is defined as

$$f(z) = (z + \tanh(z)) \sigma(z). \quad (32)$$

Közkurt et al. also proposed a parametric version which we will call parametric TanhSig (pTanhSig):

$$f(z) = (z + a \cdot \tanh(z + a)) \sigma(z), \quad (33)$$

where a is a fixed parameter [167].

⁴The author had probably a typo in the definition in the original paper [166]; we present the formula we think Kumar and Sodhi intended to write — it resembles the NewSigmoid and fits the numerical values given in the paper.

3.2.24 Multistate activation function (MSAF)

The multistate activation function (MSAF) is a logistic sigmoid based AF proposed in [168]. The general MSAF is defined as

$$f(z) = a + \sum_{k=1}^N \frac{1}{1 + \exp(-z + b_k)}, \quad (34)$$

where a and b_k , $k = 1, \dots, N$ are fixed parameters; $a \in \mathbb{R}$, $N \in \mathbb{N}^+$, $b_k \in \mathbb{R}^+$, and $b_1 < b_2 < \dots < b_N$ [168]. If $a = 0$, it is named as N -order⁵ MSAF.

There is also a special case called symmetrical MSAF (SymMSAF) defined as

$$f(z) = -1 + \frac{1}{1 + \exp(-z)} + \frac{1}{1 + \exp(-z - a)}, \quad (35)$$

where a is required to be significantly smaller than 0 [168]

3.2.25 Rootsig and others

The rootsig is one of the activations listed in [169]. It is defined as

$$f(z) = \frac{az}{1 + \sqrt{1 + a^2 z^2}}, \quad (36)$$

where a is a parameter [169]. This function is called rootsig in [137] where the authors list a variant with $a = 1$.

There are also several other unnamed sigmoids in [169]:

$$f(z) = z \frac{\operatorname{sgn}(z) z - a}{z^2 - a^2}, \quad (37)$$

$$f(z) = \frac{az}{1 + |az|}, \quad (38)$$

and

$$f(z) = \frac{az}{\sqrt{1 + a^2 z^2}}. \quad (39)$$

3.2.26 Sigmoid and tanh combinations

Guevraa et al. proposed several activations mostly combining the logistic sigmoid, tanh, and linear function in [170]. The general approach is

$$f(z) = \begin{cases} g(z), & z \geq 0, \\ h(z), & z < 0, \end{cases}, \quad (40)$$

where $g(z)$ and $h(z)$ are two different AFs [170]. The authors used the following pairs $\{g(z), h(z)\}$: $\{\sigma_2(z), \tanh(z)\}$, $\{\sigma_2(z), \tanh(z)\}$, $\{\sigma_2(z), 0\}$, $\{\tanh(z), 0\}$, $\{\sigma_2(z), az\}$, and $\{\tanh(z), az\}$, where $a > 0$ is a fixed parameter and

$$\sigma_2(z) = \frac{2}{1 + \exp(-z)} - 1. \quad (41)$$

Guevraa et al. also proposed an AF we termed SigLU (see section 3.6.52) and nonadaptive variant of PTELU.

3.3 Class of sigmoid-weighted linear units

The SiLU is the most common example of a larger class of sigmoidal units defined as

$$f(z) = z \cdot s(z), \quad (42)$$

where $s(z)$ is any sigmoidal function; it becomes the SiLU if the logistic sigmoid function is used. The SiLU is thus defined as

$$f(z) = z \cdot \sigma(z), \quad (43)$$

⁵This does not exactly fit into the exemplar MSAF of order two presented in [168]; it is possible that authors intended another constraint $b_1 = 0$ for such case.

where $\sigma(z)$ is the logistic sigmoid [171]. The SiLU has the output range of $(-0.5, \infty)$ [1] and was first used [171] for reinforcement learning tasks such as SZ-Tetris and Tetris. The SiLU was also found to work well for the CIFAR-10/100 [158] and ImageNet [172, 173] tasks in [49]. The adaptive variant of the SiLU is called swish (see section 4.4.1) [49].

For the purposes of this work, we also consider any squashing functions $s(z)$ and not necessarily only sigmoids — for example, we classify rectified hyperbolic secant (see section 3.3.27) as a member of this class. We also list functions that are closely based on the SiLU and its variants.

A similar approach named weighted sigmoid gate unit (WiG) was proposed in [174], where the AF was used only for gating each of the raw inputs:

$$f(\mathbf{x})_i = x_i \cdot \sigma(z) = x_i \cdot \sigma(\mathbf{w}_i^T \mathbf{x} + b_i), \quad (44)$$

where \mathbf{x} denotes the vector of raw inputs, \mathbf{w}_i the weights of neuron i and b_i its bias [174]

3.3.1 Gaussian error linear unit (GELU)

Gaussian error linear unit (GELU) [175] is an activation function based on the standard Gaussian cumulative distribution function, and it weights inputs by their value rather than gating them as ReLUs do [175]. It is defined as

$$f(z) = z \cdot \Phi(z) = z \cdot \frac{1}{2} \left(1 + \operatorname{erf} \left(\frac{z}{\sqrt{2}} \right) \right), \quad (45)$$

where $\Phi(z)$ is the standard Gaussian cumulative distribution function (CDF) and $\operatorname{erf}(x)$ is the Gauss error function [175]. It is similar to the SiLU but it uses $\Phi(z)$ instead of the $\sigma(z)$. However, due to the complicated formula, the GELU can be approximated as

$$f(z) = \frac{1}{2} z \left(1 + \tanh \left(\sqrt{\frac{2}{\pi}} (z + 0.044715 z^3) \right) \right) \quad (46)$$

or

$$f(z) = z \cdot \sigma(1.702z), \quad (47)$$

if the performance gains are worth the loss of exactness [175]. The function is similar to SiLU (see section 3.3), it only uses Gaussian CDF $\Phi(z)$ instead of the logistic distribution CDF $\sigma(z)$ [175]. GELU was found to outperform many competitors (e.g., ReLU, ELU, SELU, continuously differentiable exponential linear unit (CELU), sigmoid, tanh) in [176]. Hendrycks and Gimpel also proposed to parameterize the GELU by μ and σ^2 — the parameters defining mean and variance of the Gaussian distribution whose CDF is used in the GELU [175], however, only the standard Gaussian distribution was used in experiments in [175]. Replacing ReLUs with GELUs led to better performance in [177]. More details about GELU are available in [176].

3.3.2 Symmetrical Gaussian error linear unit (SGELU)

A symmetric variant of GELU called symmetrical Gaussian error linear unit (SGELU) was proposed in [178]. It is defined as

$$f(z) = a \cdot z \cdot \operatorname{erf} \left(\frac{z}{\sqrt{2}} \right), \quad (48)$$

where a is a fixed hyperparameter [178]. The symmetrical nature of the SGELU also leads to more symmetrically distributed weights of the neural network compared to SGELU [178]; it is believed that normal distribution of the weights can make the network more rational, accurate, and robust [178].

3.3.3 Cauchy linear unit (CaLU)

Another function related to the GELU and SiLU is the Cauchy linear unit (CaLU) [179] which uses the CDF of the standard Cauchy distribution instead of the Gaussian CDF in GELU and logistic sigmoid in SiLU. It is defined as

$$f(z) = z \cdot \Phi_{\text{Cauchy}}(z) = z \cdot \left(\frac{\tan^{-1}(z)}{\pi} + \frac{1}{2} \right), \quad (49)$$

where $\Phi_{\text{Cauchy}}(z)$ is the CDF of the standard Cauchy distribution [179].

3.3.4 Laplace linear unit (LaLU)

Another function related to the GELU and SiLU is the Laplace linear unit (LaLU) [179] which uses the CDF of the Laplace distribution; it is defined as

$$f(z) = z \cdot \Phi_{\text{Laplace}}(z) = z \cdot \begin{cases} 1 - \frac{1}{2} \exp(-z), & z \geq 0, \\ \frac{1}{2} \exp(z), & z < 0, \end{cases} \quad (50)$$

where $\Phi_{\text{Laplace}}(z)$ is the CDF of the Laplace distribution [179].

3.3.5 Collapsing linear unit (LaLU)

The Collapsing linear unit (CoLU) is an AF similar to the SiLU proposed in [180]. It is defined as

$$f(z) = z \cdot \frac{1}{1 - z \exp(-(z + \exp(z)))}. \quad (51)$$

3.3.6 Triple-state swish

The triple-state swish unit (TS-swish)⁶ is a cascaded AF similar to TS-sigmoid (see section 3.2.6) [154]; it is defined as

$$f(z) = \frac{z}{1 + \exp(-z)} \left(\frac{1}{1 + \exp(-z)} + \frac{1}{1 + \exp(-z + a)} + \frac{1}{1 + \exp(-z + b)} \right), \quad (52)$$

where a and b are fixed parameters [154].

3.3.7 Generalized swish

A SiLU variant called generalized swish⁷ was proposed in [154]. It is defined as

$$f(z) = z \cdot \sigma(\exp(-z)). \quad (53)$$

3.3.8 Exponential swish

Another SiLU variant called exponential swish⁸ was proposed in [154]. It is defined as

$$f(z) = \exp(-z) \sigma(z). \quad (54)$$

3.3.9 Derivative of sigmoid function

The derivative of logistic sigmoid was used as an AF in [154]. Koçak and Üstündağ Şiray formulate the AF using the following form

$$f(z) = \exp(-z) (\sigma(z))^2. \quad (55)$$

3.3.10 Gish

Gish is another SiLU variant [181]; the gish is defined as

$$f(z) = z \cdot \ln(2 - \exp(-\exp(z))). \quad (56)$$

Kaytan, Aydilek, and Yeroğlu found that gish outperformed logistic sigmoid, softplus, ReLU, LReLU, ELU, swish, mish, logish, and smish on the MNIST [182] and CIFAR-10 [158] datasets [181].

3.3.11 Logish

Logish is yet another SiLU variant [183]; it is defined as

$$f(z) = z \cdot \ln(1 + \sigma(z)). \quad (57)$$

⁶Koçak and Üstündağ Şiray called the function swish but it is actually based on the SiLU.

⁷Also based on the SiLU instead of its adaptive variant swish.

⁸Again, based on the SiLU instead of its adaptive variant swish.

3.3.12 LogLogish

LogLogish is a SiLU variant based on the LogLog (see section 3.2.20) [179]; it is defined as

$$f(z) = z \cdot (1 - \exp(-\exp(z))). \quad (58)$$

3.3.13 ExpExpish

ExpExpish is a SiLU variant [179]; it is defined as

$$f(z) = z \cdot \exp(-\exp(-z)). \quad (59)$$

3.3.14 Self arctan

The self arctan is an AF proposed in [153] whose formula resembles the SiLU. The self arctan is defined as

$$f(z) = z \cdot \tan^{-1}(z), \quad (60)$$

where $\tan^{-1}(z)$ is the arctangent function [153].

3.3.15 Parametric logish

Zhu et al. also proposed a parametric variant of logish — we will call it parametric logish (pLogish) in this work. It is defined as

$$f(z_i) = a_i z_i \cdot \ln(1 + \sigma(b_i z_i)), \quad (61)$$

where a and b are fixed parameters [183]; Zhu et al. used $a = 1$ and $b = 10$ in [183].

3.3.16 Phish

Phish is a SiLU variant combining GELU and tanh [184]; it is defined as

$$f(z) = z \cdot \tanh(\text{GELU}(z)). \quad (62)$$

The phish was found to outperform GELU, tanh, logistic sigmoid, and ReLU; it performed similarly as the mish and swish in the experiments in [184].

3.3.17 Suish

The suish [96] was proposed as an alternative to the swish AF in [185]. It is defined as

$$f(z) = \max(z, z \cdot \exp(-|z|)). \quad (63)$$

3.3.18 Tangent-sigmoid ReLU (TSReLU)

The tangent-sigmoid ReLU (TSReLU) [186] is an AF very similar to phish, mish, and TanhExp — it just uses the logistic sigmoid instead of the GELU in phish, softplus in mish, and the exponential in TanhExp. It is defined as

$$f(z) = z \cdot \tanh(\sigma(z)). \quad (64)$$

3.3.19 Tangent-bipolar-sigmoid ReLU (TBSReLU)

The tangent-bipolar-sigmoid ReLU (TBSReLU) is a variant of TSReLU proposed in [186]. It is defined as

$$f(z) = z \cdot \tanh\left(\frac{1 - \exp(-z)}{1 + \exp(-z)}\right). \quad (65)$$

3.3.20 Log-sigmoid

A logarithm of the logistic sigmoid is sometimes used as an activation function [187]. It is defined as

$$f(z) = \ln(\sigma(z)) = \ln\left(\frac{1}{1 + \exp(-z)}\right). \quad (66)$$

3.3.21 Derivative of sigmoid-weighted linear unit (dSiLU)

The derivative of sigmoid-weighted linear unit (dSiLU) can also be used as an activation function resembling a sigmoid [171]. It is defined as

$$f(z) = \sigma(z) (1 + z (1 - \sigma(z))), \quad (67)$$

where $\sigma(z)$ is the logistic sigmoid [171]. The dSiLU has a maximum value of around 1.1, and the minimum is approximately -0.1 [171].

3.3.22 Double sigmoid-weighted linear unit (DoubleSiLU)

The double sigmoid-weighted linear unit (DoubleSiLU)⁹ is an AF proposed in [188]. It is defined as

$$f(z) = z \cdot \frac{1}{1 + \exp\left(-z \cdot \frac{1}{1 + \exp(-z)}\right)}, \quad (68)$$

where $\sigma(z)$ is the logistic sigmoid [188].

3.3.23 Modified sigmoid-weighted linear unit (MSiLU)

A modified sigmoid-weighted linear unit (MSiLU) is a variant of the SiLU that has faster convergence than the SiLU [189]. It is defined as

$$f(z) = z \cdot \sigma(z) + \frac{\exp(-z^2 - 1)}{4}, \quad (69)$$

where $\sigma(z)$ is the logistic sigmoid [189].

3.3.24 Hyperbolic tangent sigmoid-weighted linear unit (TSiLU)

Another SiLU variant is the hyperbolic tangent sigmoid-weighted linear unit (TSiLU) [188], which combines the tanh and SiLU. It is defined¹⁰ as

$$f(z) = \frac{\exp\left(\frac{z}{1 + \exp(-z)}\right) - \exp\left(-\frac{z}{1 + \exp(-z)}\right)}{\exp\left(\frac{z}{1 + \exp(-z)}\right) + \exp\left(-\frac{z}{1 + \exp(-z)}\right)}. \quad (70)$$

3.3.25 Arctan sigmoid-weighted linear unit (ASiLU)

Arctan sigmoid-weighted linear unit (ATSiLU) is yet another SiLU variant proposed in [188]; it is defined as

$$f(z) = \tan^{-1}\left(z \cdot \frac{1}{1 + \exp(-z)}\right). \quad (71)$$

3.3.26 SwAT

Verma, Chug, and Singh proposed an AF named SwAT combining the SiLU and arctan in [188]. This function is defined as

$$f(z) = z \cdot \frac{1}{1 + \exp(-\tan^{-1} \text{left}(z))}. \quad (72)$$

3.3.27 Rectified hyperbolic secant

A rectified hyperbolic secant activation function was proposed in [190]. This function is totally differentiable, symmetric about the origin, and is approaching zero for inputs going to positive or negative infinity:

$$f(z) = z \cdot \text{sech}(z), \quad (73)$$

where $\text{sech}(z)$ is the hyperbolic secant function [190].

⁹Verma, Chug, and Singh termed the unit as DSiLU but that would collide with the dSiLU (see section 3.3.21) proposed earlier by Elfwing, Uchibe, and Doya.

¹⁰The formula in [188] was wrong as it evaluated to $\frac{2x}{0}$, we present the formula we think authors intended.

3.3.28 Linearly scaled hyperbolic tangent (LiSHT)

A linearly scaled hyperbolic tangent (LiSHT) activation function was proposed in [191] to address the problem of vanishing gradients and the non-utilization of large negative input values. The LiSHT function is defined as

$$f(z) = z \cdot \tanh(z). \quad (74)$$

The output range of LiSHT function is $[0, \infty]$ [1]. The output of LiSHT is close to the ReLU (see section 3.6) and swish for large positive values [191]; however, unlike the aforementioned AFs, the output is symmetric, and, therefore, it behaves identically for large negative values. While the LiSHT is symmetric, the fact that its output is unbounded and non-negative could be considered a disadvantage [1]. The effectiveness of the LiSHT activation function was tested on several different architectures ranging from multilayer perceptron (MLP) and residual neural networks to LSTM-based networks and on various tasks — the Iris dataset, the MNIST [182], CIFAR-10 and CIFAR-100 [158] and the *sentiment140* dataset from Twitter [192, 193] for sentiment analysis [191].

A parametric version of LiSHT named SoftModulusT (see section 3.6.31) was proposed in [194].

3.3.29 Mish

A popular activation function mish [195] is a combination of the tanh and softplus activation function; the function resembles swish activation (see section 4.4.1). It is defined as

$$f(z) = z \cdot \tanh(\text{softplus}(z)) = z \cdot \tanh(\ln(1 + \exp(z))). \quad (75)$$

Mish was found to outperform swish; it performed similarly to $f(z) = z \cdot \ln(1 + \tanh(\exp(z)))$ but this activation function was found to often lead to unstable training [195]. The mish was found to outperform swish and ReLU for many architectures such as various ResNet architectures [196], Inception v3 [197], DenseNet-121 [198], and others [195]. Detailed comparison with other activation functions was run using the Squeeze Net [199] where it outperformed swish, GELU, ReLU, ELU, LReLU, SELU, softplus, S-shaped ReLU (SReLU), inverse square root unit (ISRU), tanh, and randomized leaky ReLU (RReLU) [195]. The mish activation function was, for example, used in the YOLOv4 [200] and its variant Scaled-YOLOv4 [201].

3.3.30 Smish

The smish [202] is a variant of the mish where the exponential function is replaced by the logistic sigmoid. It is, therefore, defined as

$$f(z) = az \cdot \tanh(\ln(1 + \sigma(bz))), \quad (76)$$

where a and b are parameters [202]; however, Wang, Ren, and Wang recommend $a = 1$ and $b = 1$ based on a small parameter search in [202].

3.3.31 TanhExp

Similarly as the mish is the combination of tanh and softplus, the TanhExp [203] is a combination of tanh and the exponential function [203, 204]. It is defined as

$$f(z) = z \cdot \tanh(\exp(z)). \quad (77)$$

3.3.32 Serf

The serf is an AF similar to the mish; however, it uses the error function instead of the tanh [205]. It is defined as

$$f(z) = z \cdot \text{erf}(\ln(1 + \exp(z))), \quad (78)$$

where erf is the Gauss error function [205]. It was found to outperform mish, GELU, and ReLU for various architectures on Multi30K [206], ImageNet [172, 173], the CIFAR-10, and CIFAR-100 [158] datasets; see [205] for details.

3.3.33 Efficient asymmetric nonlinear activation function (EANAF)

An activation function combining tanh and softplus called efficient asymmetric nonlinear activation function (EANAF) was proposed in [207]. The function is defined as

$$f(z) = z \cdot g(h(z)), \quad (79)$$

where $h(z)$ is the softplus function and $g(z) = \tanh\left(\frac{z}{2}\right)$, which can be simplified to

$$f(z) = \frac{z \cdot \exp(z)}{\exp(z) + 2}. \quad (80)$$

The EANAF is continuously differentiable. The EANAF is very similar to swish with similar amount of computation but Chai et al. found that it performs better than swish and several other activation functions in RetinaNet [208] and YOLOv4 [201] architectures on object detection tasks [207].

3.3.34 SinSig

SinSig [209] is a self-gated non-monotonic activation function defined as

$$f(z) = z \cdot \sin\left(\frac{\pi}{2} \sigma(z)\right), \quad (81)$$

where $\sigma(z)$ is the logistic sigmoid function [209]. While SinSig is similar to swish and mish, it outperformed them in experiments in [209] as the number of layers in a neural network increased. It was also shown that the SinSig converges faster. The SinSig outperformed ReLU and mish on several deep architectures including ResNet 20 v2 [210], ResNet 110 v2 [210], SqueezeNet [211], and ShuffleNet [212] among others on the CIFAR-100 task [158] in experiments in [209].

3.3.35 Gaussian error linear unit with sigmoid activation function (SiELU)

The with sigmoid activation function (SiELU) was proposed in [213]; it is defined as

$$f(z) = z \sigma\left(2\sqrt{\frac{2}{\pi}}(z + 0.044715z^3)\right). \quad (82)$$

3.4 Gated linear unit (GLU)

A gated activation called gated linear unit (GLU) similar to SiLU (see section 3.3) for use in recurrent neural networks (RNNs) was proposed in [214]. The GLU is defined as

$$f(z, z') = z \otimes \sigma(z'), \quad (83)$$

where \otimes is the element-wise product and z and z' are two learned linear transformations of input vector x [215, 216].

3.4.1 Gated tanh unit (GTU)

A gated activation called gated tanh unit (GTU) similar to GLU (see section 3.4) for use in RNNs was proposed in [217]. The GTU is defined as

$$f(z, z') = \tanh(z) \otimes \sigma(z'), \quad (84)$$

where \otimes is the element-wise product and z and z' are two learned linear transformations of input vector x [215].

3.4.2 Gated ReLU (ReGLU)

Another GLU extension is the gated (ReGLU) [214, 215]. The ReGLU is defined as

$$f(z, z') = z \otimes \text{ReLU}(z'), \quad (85)$$

where \otimes is the element-wise product and z and z' are two learned linear transformations of input vector x [215].

3.4.3 Gated GELU (GEGLU)

A GELU-based GLU extension is the gated (GEGLU) [215]; it is defined as

$$f(z, z') = z \otimes \text{GELU}(z'), \quad (86)$$

where \otimes is the element-wise product and z and z' are two learned linear transformations of input vector x [215].

3.4.4 Swish GELU (SwiGLU)

A swish-based GLU extension is the gated swish (SwiGLU) [215]; it is defined as

$$f(z, z') = z \otimes \text{swish}(z'), \quad (87)$$

where \otimes is the element-wise product, z and z' are two learned linear transformations of input vector x , and swish is the swish with its own trainable parameter [215].

3.5 Softmax

The softmax is not a usual type of AF taking in a single value, but it takes all the output value of the unit i and, also, the output values of other units in order to compute a soft argmax of the values. It is defined as

$$f(z_j) = \frac{\exp(z_j)}{\sum_{k=1}^N \exp(z_k)}, \quad (88)$$

where $f(z_j)$ is the output of a neuron j in a softmax layer consisting of N neurons [218, 219].

3.5.1 β -softmax

The β -softmax is a softmax extension proposed in [220]; it is defined as

$$f(z_j) = \frac{\int \exp(bz_j)}{\sum_{k=1}^N \int \exp(bz_k)}, \quad (89)$$

where $f(z_j)$ is the output of a neuron j in a softmax layer consisting of N neurons and b takes random value from \mathbb{N}^{+11} [220].

3.6 Rectified linear function (ReLU)

The rectified linear unit (ReLU) [221] is widely regarded as the most popular activation function in modern feedforward networks [136, 222, 223] due to its simplicity and improved performance [1]. It has been observed that ReLUs can significantly expedite the convergence of stochastic gradient descent [224]. Additionally, traditional ReLUs are computationally less expensive compared to activation functions like the logistic or tanh functions [223]. ReLUs often outperform sigmoidal activation functions [222]. However, a drawback of ReLUs is the potential for neurons to become "dead" or "disabled" during training. This means that they may never activate again for any input, resulting in a permanently zero output gradient [223]. This issue can occur after a weight update when a large gradient flows through the unit [223]. This might happen after a weight update after a large gradient flows through the unit [223]. However, ReLUs often lead to faster convergence than for sigmoid activation, as shown in [225]. It can also be shown that ReLUs and rational function efficiently approximate each other [226]. The ReLU was used as an example of the more general class of piecewise affine AFs for neural network verification¹² using theorem provers in [227].

A ReLU is mathematically defined as the maximum of zero and the input value [222, 228]:

$$f(z) = \max(0, z). \quad (90)$$

ReLU is commonly recommended as the default choice for feedforward networks due to its usually superior performance compared to sigmoidal functions and its computational efficiency [223]; furthermore, it works comparably to its modifications [228]. Many popular NN models utilize ReLU as the activation function of choice, e.g., [224, 229].

Many ReLU modification and derivations were proposed [228, 230] — e.g. leaky ReLU (LReLU) [231], very leaky ReLU (VReLU) [232], parametric ReLU [233], randomized leaky ReLU (RReLU) [234] or S-shaped ReLU [235]. Smoothed modifications are, for example, exponential linear unit [236] and softplus [222]. Most of the modifications solve the problem of dying out neurons as they allow for gradient flows for any input.

3.6.1 Shifted ReLU

A Shifted ReLU [236] is a simple translation of a ReLU and is defined as

$$f(z) = \max(-1, z). \quad (91)$$

¹¹No further specification was provided in [220].

¹²More details are out of the scope of this work, see [227] for more details.

3.6.2 Leaky ReLU (LReLU)

Leaky ReLU (LReLU) [231] is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{z}{a}, & z < 0, \end{cases} \quad (92)$$

where $a \in (1, \infty)$ is set to large number;¹³ the recommended setting from [231] is $a = 100$.

LReLU solves the problem of dying neurons when neurons have permanently zero output gradient in classical ReLU by "leaking" the information for $z < 0$ instead of outputting exact zero. Both ReLU and LReLU can be considered to be a special case of the maxout unit (see section 4.47) [1]. A theoretical analysis of the ReLU and LReLU is available in [237].

Very leaky ReLU (VLRReLU) [232] is almost identical to the LReLU but has much higher slope when the z is negative for faster training [232] by setting $a_i = 3$. While it can be considered as a special case of LReLU, some researchers consider it as a separate case, e.g., [230].

The so-called optimized leaky ReLU (OLReLU) [238] propose another reformulation of LReLU and calculation of the slope parameter a that is inspired by the RReLU (see section 3.6.3):

$$f(z) = \begin{cases} z, & z \geq 0, \\ z \cdot \exp(-a), & z < 0, \end{cases} \quad (93)$$

where

$$a = \frac{u + l}{u - l}, \quad (94)$$

where u and l are hyperparameters of the bounds of the RReLU [238].

3.6.3 Randomized leaky ReLU (RReLU)

RReLU is a leaky ReLU where the leakiness is stochastic during the training [234], i.e.:

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ \frac{z_i}{a_i}, & z_i < 0, \end{cases} \quad (95)$$

where a_i is a sampled for each epoch and neuron i from the uniform distribution: $a_i \sim U(l, u)$ where $l < u$ and $l, u \in (0, \infty)$ [234]. Similarly as in the dropout approach [239], an average over all a_i over is taken during inference phase — the a_i is set to $\frac{l+u}{2}$:

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ \frac{z_i}{\frac{l+u}{2}}, & z_i < 0. \end{cases} \quad (96)$$

Recommended distribution is $U(3, 8)$ for sampling the a_i [234].

3.6.4 Softsign randomized leaky ReLU (S-RReLU)

The softsign randomized leaky ReLU (S-RReLU)¹⁴ is a RReLU combined with the softsign proposed in [240, 241]. It is defined as

$$f(z_i) = \begin{cases} \frac{1}{(1+z_i)^2} + z_i, & z_i \geq 0, \\ \frac{1}{(1+z_i)^2} + a_i z_i, & z_i < 0, \end{cases} \quad (97)$$

where a_i is a sampled for each epoch and neuron i from the uniform distribution: $a_i \sim U(l, u)$ where $l < u$ and $l, u \in (0, \infty)$ [240]. Elakkiya and Dejeu used $l = \frac{1}{8}$ and $u = \frac{1}{3}$ [240].

¹³Depending on the source, researchers use either this form $\frac{z}{a}$ or the inverted form az for the negative inputs.

¹⁴Elakkiya and Dejeu used S-RReLU as a name and not an abbreviation; however, since S-RReLU is a combination of the softsign and RReLU, we feel that using it as an abbreviation is appropriate.

3.6.5 Sloped ReLU (SReLU)

A Sloped ReLU (SReLU) [242] is similar to the LReLU — whereas the LReLU parameterizes the slope for negative inputs, the SReLU parameterizes the slope of ReLU for positive inputs. It is, therefore, defined as

$$f(z) = \begin{cases} az, & z \geq 0, \\ 0, & z < 0, \end{cases} \quad (98)$$

where a is a fixed, predetermined parameter [242]. Seo, Lee, and Kim recommended $a \in [1, 10]$ based on their experiments in [242].

3.6.6 Noisy ReLU (NReLU)

A stochastic variant of the ReLU called noisy ReLU (NReLU) was proposed in [221]:

$$f(z) = \max(0, z + a), \quad (99)$$

where a is a stochastic parameter $a \sim N(0, \sigma(z))$, $N(0, \sigma^2)$ is the Gaussian distribution with zero mean and variance σ^2 and $\sigma(z)$ is the standard deviation of the inputs z . The NReLU was designed for use with Restricted Boltzmann machines [221]. More details about the NReLU is available in [221].

3.6.7 SineReLU

The SineReLU [243, 244] is a ReLU based activation that uses trigonometric functions for negative inputs. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ a(\sin(z) - \cos(z)), & z < 0, \end{cases} \quad (100)$$

where a is a fixed parameter [243, 244].

3.6.8 Minsin

The minsin is a ReLU-based AF used in [96]. It is defined as

$$f(z) = \min(z, \sin(z)) = \begin{cases} \sin(z), & z \geq 0, \\ z, & z < 0. \end{cases} \quad (101)$$

3.6.9 Variational linear unit (VLU)

The variational linear unit (VLU) is an AF combining the ReLU and sine functions proposed in [243]. It is defined as

$$f(z) = \text{ReLU}(z) + a \sin(bz) = \max(0, z) + a \sin(bz), \quad (102)$$

where a and b are fixed parameters [243].

3.6.10 Spatial context-aware activation (SCAA)

The spatial context-aware activation (SCAA) is a ReLU extension proposed in [245]. The ReLU performs an element-wise max operation on the feature map \mathbf{X} :

$$\text{ReLU}(\mathbf{X}) = \max(\mathbf{X}, \mathbf{0}), \quad (103)$$

where $\text{ReLU}(\mathbf{X})$ is the ReLU in the matrix notation and $\mathbf{0}$ is a matrix of zeroes with the same shape as \mathbf{X} [245]. The SCAA first applies a depth-wise convolution on \mathbf{X} to produce spatial context aggregated feature map denoted $\mathbf{f}_{\text{DW}}(\mathbf{X})$ and then proceeds with the elementwise max operation [245]; the SCAA is, therefore, defined as

$$\mathbf{f}(\mathbf{X}) = \max(\mathbf{X}, \mathbf{f}_{\text{DW}}(\mathbf{X})). \quad (104)$$

3.6.11 Randomly translational ReLU (RT-ReLU)

A randomly translational ReLU (RT-ReLU) is a ReLU with a randomly added jitter during each iteration of the training process [246]. It is defined as

$$f(z_i) = \begin{cases} z_i + a_i, & z_i + a_i \geq 0, \\ 0, & z_i + a_i < 0, \end{cases} \quad (105)$$

where a_i is stochastic parameter for each neuron i randomly sampled from the Gaussian distribution at each iteration, $a_i \sim N(0, \sigma^2)$, where σ^2 is the variance of the Gaussian distribution. The authors Cao et al. set the $\sigma^2 = 0.75^2$ for their experiments [246]. The a_i is set to 0 during the test phase [1].

3.6.12 Natural-Logarithm-ReLU (NLReLU)

The natural-logarithm-ReLU (NLReLU) introduces non-linearity to ReLU similarly as rectified linear tanh (ReLUtanh) (see section 4.2.36) but only for positive part of the activation function [1]:

$$f(z) = \ln(a \cdot \max(0, z) + 1), \quad (106)$$

where a is a predefined constant [247].

3.6.13 Softplus linear unit (SLU)

An activation function softplus linear unit (SLU) combining the ReLU with the softplus activation function was proposed in [248]; the function is based around the assumption that zero mean activations improve learning performance [248]. The SLU is defined as

$$f(z) = \begin{cases} az, & z \geq 0, \\ b \ln(\exp(z) + 1) - c, & z < 0, \end{cases} \quad (107)$$

where a_i , b_i , and c_i are predefined parameters; however, to ensure that the function is continuous, differentiable at zero and to avoid vanishing or exploding gradients, its parameters are set to $a = 1$, $b = 2$, and $c = 2 \ln(2)$ [248]. The SLU is therefore equal to

$$f(z) = \begin{cases} z, & z \geq 0, \\ 2 \ln \frac{\exp(z)+1}{2}, & z < 0. \end{cases} \quad (108)$$

3.6.14 Rectified softplus (ReSP)

Another activation function combining ReLU and softplus called rectified softplus (ReSP) [1] was proposed in [249]. The function is defined as

$$f(z) = \begin{cases} az + \ln(2), & z \geq 0, \\ \ln(1 + \exp(z)), & z < 0, \end{cases} \quad (109)$$

where a is a fixed hyperparameter controlling the slope [249]. Larger values of a between 1.4 and 2.0 were found to work well [249].

3.6.15 Parametric rectified non-linear unit (PReLU)

A ReLU variant called parametric rectified non-linear unit (PReLU) [250] replaces the linear part of the ReLU for positive inputs by a non-linear function similarly to RePU (see section 3.6.39). It is defined as

$$f(z) = \begin{cases} z - a \cdot \ln(z + 1), & z \geq 0, \\ 0, & z < 0, \end{cases} \quad (110)$$

where a is a fixed hyperparameter [250] — however, this parameter could be adaptive similarly as in PReLU (see section 4.2.1) that PReLU extends since Jaafari, Ellahyani, and Charfi thought of the PReLU as non-adaptive function for some reason [250].

3.6.16 Bounded ReLU (BReLU)

A BReLU [251] is a variant of ReLU that limits the output as the unlimited output of the original ReLU might lead to an instability [1]. It is defined as

$$f(z) = \min(\max(0, z), a) = \begin{cases} 0, & z \leq 0, \\ z, & 0 < z < a, \\ a, & z > a, \end{cases} \quad (111)$$

where a is a predefined parameter [251]. The BReLU appeared later in the literature under the name *ReLU_N* in [252], where it seems that it was independently proposed.

3.6.17 Hard sigmoid

A Hard sigmoid is very similar to BReLU; it is a very crude approximation of the logistic sigmoid and is commonly defined [101, 253] as

$$f(z) = \max\left(0, \min\left(\frac{z+1}{2}, 1\right)\right). \quad (112)$$

Other definitions are sometimes used; e.g. variant from [254] is defined as

$$f(z) = \max(0, \min(0.2z + 0.5, 1)). \quad (113)$$

While the Hard sigmoid is not as commonly used as the logistic sigmoid, it can be used, for example, in binarized neural network with stochastic activation functions [253] — the binaryized neural networks can lead to much faster inference than regular neural networks, e.g., Courbariaux et al. reached up to $7\times$ speed up without any loss in classification accuracy [253] (however, even better speed-ups can be obtained using, for example, field-programmable gate array (FPGA) implementations as in [255]).

3.6.18 HardTanh

The HardTanh is another piecewise linear function; it is very similar to Hard sigmoid, but it approximates the tanh instead of the logistic sigmoid. It is defined as

$$f(z) = \begin{cases} a, & z < a, \\ z, & a \leq z \leq b, \\ b, & z > b, \end{cases} \quad (114)$$

where a and b are fixed parameters [256]; Liu et al. used $a = -1$ and $b = 11$ in [257]. NNs with HardTanh are more suitable for linear predictive control than NNs with ReLUs as they usually require less hidden layers and neurons for representing identical min-max maps [256].

3.6.19 Shifted HardTanh

Kim et al. proposed HardTanh variants with vertical and horizontal shifts in [258]. The SvHardTanh¹⁵ is defined as

$$f(z) = \begin{cases} -1 + a, & z < -1, \\ z + a, & -1 \leq z \leq 1, \\ 1 + a, & z > 1, \end{cases} \quad (115)$$

where a is a fixed parameter [258]. Kim et al. used HardTanh variant with thresholds -1 and 1 ; a more general variant with parametric thresholds from eq. (114) could be defined similarly.

The SvHardTanh is defined as

$$f(z) = \begin{cases} -1 + a, & z < -1, \\ z + a, & -1 \leq z \leq 1, \\ 1 + a, & z > 1, \end{cases} \quad (116)$$

where a is a fixed parameter [258].

The ShHardTanh is defined as

$$f(z) = \begin{cases} -1, & z < -1 - a, \\ z, & -1 - a \leq z \leq 1 - a, \\ 1, & z > 1 - a, \end{cases} \quad (117)$$

where a is a fixed parameter [258].

Kim et al. used HardTanh variant with thresholds -1 and 1 ; more general variants of SvHardTanh and ShHardTanh with parametric thresholds from eq. (114) could be defined similarly.

3.6.20 Hard swish

A linearized variant of the swish AF (see section 4.4.1) was proposed in [259]. It is defined as

$$f(z) = z \cdot \begin{cases} 0, & z \leq -3, \\ 1, & z \geq 3, \\ \frac{z}{6} + \frac{1}{2}, & -3 < z < 3. \end{cases} \quad (118)$$

The linearization allows for more efficient computation [259].

¹⁵Both SvHardTanh and ShHardTanh are named using the same convention as shifted ELUs (see section 4.2.56) for the purposes of this work.

3.6.21 Truncated rectified (TRec) activation function

The truncated rectified (TRec) AF is a truncated variant of the ReLU [260]. It resembles onesided variant of the Hardshrink (see section 3.6.22) — it is defined as

$$f(z) = \begin{cases} z, & z > a, \\ 0, & z \leq a, \end{cases} \quad (119)$$

where a is a fixed parameter. Konda, Memisevic, and Krueger used $a = 1$ for most of their experiments [260].

3.6.22 Hardshrink

The Hardshrink [67, 260–262] (named *thresholded linear* AF in [260]¹⁶) is very similar to Hard sigmoid, TRec, and other piecewise linear functions; it is defined as

$$f(z) = \begin{cases} z, & z > a, \\ 0, & -a \leq z \leq a, \\ z, & z < -a, \end{cases} \quad (120)$$

where $a > 0$ is a fixed parameter.

3.6.23 Softshrink

The Softshrink is an AF similar to the Hardshrink used in [111, 263]. It is defined as

$$f(z) = \begin{cases} z - a, & z > a, \\ 0, & -a \leq z \leq a, \\ z + a, & z < -a, \end{cases} \quad (121)$$

where $a > 0$ is a fixed thresholding parameter [263].

3.6.24 Bounded leaky ReLU (BLReLU)

Similarly as the BReLU is a bounded variant of the ReLU, the bounded leaky ReLU (BLReLU) is a bounded variant of LReLU (see section 3.6.2) [251]. It is defined as

$$f(z) = \begin{cases} az, & z \leq 0, \\ z, & 0 < z < b, \\ az + c, & z > b, \end{cases} \quad (122)$$

where a and b are predefined parameters and c is computed such that $b = ab + c$ [251], i.e. $c = (1 - a)b$. The parameter a controls the leakiness, the parameter b is the threshold of saturation, and c is computed such that the function is continuous.

3.6.25 V-shaped ReLU (vReLU)

A V-shaped variant of ReLU called V-shaped ReLU (vReLU) is proposed in [264, 265] and tackles the problem of dying neurons that is present with ReLUs [264]. The vReLU is identical to the absolute value function and is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ -z, & z < 0. \end{cases} \quad (123)$$

The output range of vReLU is $[0, \infty)$ [1]. The *modulus* activation function later proposed in the literature by Vallés-Pérez et al. in [194] is identical to the vReLU. The absolute value function was used as an AF also in [266].

3.6.26 Pan function

The pan function is an AF similar to the vReLU and Softshrink [267, 268]. It is defined as

$$f(z) = \begin{cases} z - a, & z \geq a, \\ 0, & -a < z < a, \\ -z - a, & z \leq -a, \end{cases} \quad (124)$$

where a is a fixed boundary parameter [267].

¹⁶Konda, Memisevic, and Krueger proposed it as a novel AF but it was already proposed in [261].

3.6.27 Absolute linear unit (AbsLU)

The absolute linear unit (AbsLU) [269] is a ReLU-based AF similar to the vReLU. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ a \cdot |z|, & z < 0, \end{cases} \quad (125)$$

where $a \in [0, 1]$ is a fixed hyperparameter [269].

3.6.28 Mirrored rectified linear unit (mReLU)

The mirrored rectified linear unit (mReLU) is a bounded AF that suppresses the output for unusual inputs [270]. It is defined as

$$f(z) = \min(\text{ReLU}(1 - z), \text{ReLU}(1 + z)) = \begin{cases} 1 + z, & -1 \leq z \leq 0, \\ 1 - z, & 0 < z \leq 1, \\ 0, & \text{otherwise} . \end{cases} \quad (126)$$

3.6.29 Leaky single-peaked triangle linear unit (LSPTLU)

An AF similar to vReLU, AbsLU, and tent activation named leaky single-peaked triangle linear unit (LSPTLU) was proposed in [271]. It is defined as

$$f(z) = \begin{cases} 0.2z, & z < 0, \\ z, & 0 \leq z \leq a, \\ 2a - z, & a < z \leq 2a, \\ 0, & z \geq 2a, \end{cases} \quad (127)$$

where a is a fixed parameter [271]. An identical AF was proposed under the name leaky rectified triangle linear unit (LRTL) in [272].

3.6.30 SoftModulusQ

The SoftModulusQ is a quadratic approximation of the vReLU proposed in [194]. The SoftModulusQ is defined as

$$f(z) = \begin{cases} z^2 (2 - |z|), & |z| \geq 1, \\ |z|, & |z| < 1. \end{cases} \quad (128)$$

3.6.31 SoftModulusT

While the SoftModulusQ (see section 3.6.30) is a quadratic approximation of the vReLU (see section 3.6.25), the SoftModulusT [194] is a tanh based approximation of the vReLU. It is basically a parametric version of the LiSHT activation function (see section 3.3.28):

$$f(z) = z \cdot \tanh\left(\frac{z}{a}\right), \quad (129)$$

where a is a predetermined parameter; the authors Vallés-Pérez et al. used $a = 0.01$ in their experiments [194]. When $a = 1$, the SoftModulusT becomes the LiSHT activation function.

3.6.32 SignReLU

The combination of ReLU and softsign resulted in SignReLU [273] that improves the convergence rate and alleviates the vanishing gradient problem [273]. The SignReLU is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ a \frac{z}{|z|+1}, & z < 0, \end{cases} \quad (130)$$

where a is a fixed parameter [273, 274]; the SignReLU becomes ReLU for $a = 0$. The SignReLU was independently proposed under the name DLU in [275];¹⁷ this name is sometimes used in the literature — e.g., [267].

¹⁷[275] is a preprint of [274].

3.6.33 Li-ReLU

Elakkiya and Dejeey proposed a combination of a linear function and the ReLU in [240]; they named the function Li-ReLU¹⁸ and it is defined as

$$f(z) = \begin{cases} az + z, & z_i \geq 0, \\ az, & z_i < 0, \end{cases} \quad (131)$$

where a_i is a fixed parameter [240].

3.6.34 Concatenated ReLU (CReLU)

A concatenated ReLU (CReLU) is an adaptation of the ReLU function proposed based on the observation that filters in convolutional neural networks (CNNs) in the lower layers form pairs consisting of filters with opposite phase [276]. The CReLU conserves both negative and positive linear responses after convolution by concatenating the output of two ReLUs (hence the name) [276]. The CReLU is a function $\mathbb{R} \rightarrow \mathbb{R}^2$ and is defined as [276]

$$\mathbf{f}(z) = \begin{bmatrix} \text{ReLU}(z) \\ \text{ReLU}(-z) \end{bmatrix}, \quad (132)$$

with the output range of $[0, \infty)$ for both output elements [1].

3.6.35 Negative CReLU (NCReLU)

A CReLU extension named negative CReLU (NCReLU) was proposed in [277]; while it is very similar to CReLU, it multiplies the second element by -1 :

$$\mathbf{f}(z) = \begin{bmatrix} \text{ReLU}(z) \\ -\text{ReLU}(-z) \end{bmatrix}. \quad (133)$$

Very similar AF was proposed concurrently in [278] under the name bipolar activation function (BAF). Unlike the NCReLU, it does not produce a vector output but is applied in an alternating manner similar to All-ReLU (see section 4.34) but for neurons instead of layers. It is defined for the i -th neuron as

$$f(z_i) = \begin{cases} g(z_i), & i \% 2 = 0, \\ -g(-z_i), & i \% 2 = 1, \end{cases} \quad (134)$$

where $g(z_i)$ is any ReLU family AF and $\%$ is the modulo operation.

3.6.36 DualReLU

Where CReLU activation functions takes a single value and outputs a vector of two values, the DualReLU [279] takes two values as an input and outputs a single value. The DualReLU is a two-dimensional activation function meant as a replacement of the tanh activation function for Quasi-Recurrent neural networks [279]. It is defined as

$$f(z, z') = \max(0, z) - \max(0, z') = \begin{cases} 0, & z \leq 0 \wedge z' \leq 0, \\ z, & z > 0 \wedge z' \geq 0, \\ -b, & z \leq 0 \wedge z' > 0, \\ a - b, & z > 0 \wedge z' > 0. \end{cases} \quad (135)$$

3.6.37 Orthogonal permutation liner unit

The orthogonal permutation liner unit (OPLU) is not applied to a single neuron but always to a pair of neurons [280]. First, the neurons are grouped into pairs of neurons $\{i, j\}$ and the OPLU takes two inputs z_i and z_j of neurons i and j and produces the output

$$f(z_i, z_j) = \max(z_i, z_j) \quad (136)$$

for neuron i and

$$f(z_i, z_j) = \min(z_i, z_j) \quad (137)$$

for neuron j [280].

¹⁸Not an abbreviation.

3.6.38 Elastic ReLU (EReLU)

Another extension is the elastic ReLU (EReLU), which slightly randomly changes the slope of the positive part of the ReLU during the training [281]. The EReLU is defined as

$$f(z_i) = \begin{cases} k_i z_i, & z_i \geq 0, \\ 0, & z_i < 0, \end{cases} \quad (138)$$

where k_i is a sampled for each epoch and neuron i from the uniform distribution: $a_i \sim U(1 - \alpha, 1 + \alpha)$ where $\alpha \in (0, 1)$ is a parameter controlling the degree of response fluctuations [281]. The EReLU thus complements the principle of RReLU, which randomly changes the leakiness during the training while keeping the positive part fixed, while the EReLU changes the positive part and keeps the output constantly zero for negative inputs. The EReLU sets the k_i its expected value $E(k_i)$ which is equal to one — the EReLU becomes the ReLU during the test phase [281].

3.6.39 Power activation functions & rectified power units (RePU)

A power activation function extending ReLU together with a training scheme for better generalization was proposed in [282]. This activation function was later independently proposed under the name RePU in [283]. The RePU is defined as

$$f(z) = \begin{cases} z^a, & z \geq 0, \\ 0, & z < 0, \end{cases} \quad (139)$$

where a is a fixed parameter [282, 284]. The RePU is a generalization of several activation functions — it becomes the Heaviside step function for $a = 0$ and ReLU for $a = 1$; the case $a = 2$ is called *rectified quadratic unit* (ReQU) in [283] and *squared ReLU* in [285]; finally, the case $a = 3$ is called *rectified cubic unit* (ReCU) [283]. The disadvantage of RePU is its unbounded and asymmetric nature and that it is prone to vanishing gradient [1]. Theoretical analysis of the RePU is available in [284].

However, Berradi recommends alternating using $a = b$ and $a = \frac{1}{b}$ each epoch; i.e.:

$$f_1(z) = \begin{cases} z^b, & z \geq 0, \\ 0, & z < 0, \end{cases} \quad (140)$$

and

$$f_2(z) = \begin{cases} z^{\frac{1}{b}}, & z \geq 0, \\ 0, & z < 0. \end{cases} \quad (141)$$

Then the activation function $f_1(z)$ is used during odd epochs and $f_2(z)$ during even epochs; their mean is used during the test phase [282]. The value $b > 1$ was used in the experiments in [282] - $b \in \{1.05, 1.1, 1.15, 1.20, 1.25\}$.

3.6.40 Approximate ReLU (AppReLU)

The approximate ReLU (AppReLU)¹⁹ [286, 287] is the RePU with additional scaling parameter; it is defined as

$$f(z) = \begin{cases} a z^b, & z \geq 0, \\ 0, & z < 0. \end{cases} \quad (142)$$

3.6.41 Power linear activation function (PLAF)

The power linear activation function (PLAF)²⁰ is a class of two similar AFs proposed in [288]. The first, even power linear activation function (EPLAF), is defined as

$$f(z) = \begin{cases} z - (1 - \frac{1}{d}), & z \geq 1, \\ \frac{1}{d} |z|^d, & -1 \leq z < 1, \\ -z - (1 - \frac{1}{d}), & z < -1, \end{cases} \quad (143)$$

¹⁹Saha et al. used the abbreviation AReLU but this is already used for the Attention-based ReLU in this work.

²⁰Originally, Nasiri and Ghiasi-Shirazi named PLAF as *PowerLinear* AF. Also, its variants EPLAF and OPLAF were named as *EvenPowLin* and *OddPowLin* in [288].

where d is a fixed parameter [288]. Similarly, the second AF — odd power linear activation function (OPLAF) — is defined as

$$f(z) = \begin{cases} z - (1 - \frac{1}{d}), & z \geq 1, \\ \frac{1}{d} |z|^d, & 0 \leq z < 1, \\ -\frac{1}{d} |z|^d, & -1 \leq z < 0, \\ -z - (1 - \frac{1}{d}), & z < -1, \end{cases} \quad (144)$$

where d is a fixed parameter [288]. Nasiri and Ghiasi-Shirazi focused on the EPLAF in their work [288] and showed that EPLAF with $d = 2$ performed similarly as the ReLU for some of the tasks but it performed significantly better for other tasks; the OPLAF was not experimentally validated in [288].

3.6.42 Average biased ReLU (ABReLU)

Similarly as the RT-ReLU (see section 3.6.11), the average biased ReLU (ABReLU) [289] uses horizontal shifting in order to handle negative values [1]. It is defined as

$$f(z_i) = \begin{cases} z_i - a_i, & z_i - a_i \geq 0, \\ 0, & z_i - a_i < 0, \end{cases} \quad (145)$$

where a_i is the average of input activation map to the neuron/filter i [1, 289], which makes the function data dependent and adjusts the threshold based on the positive and negative data dominance [289]. The output range is $[0, \infty)$ [1].

3.6.43 Delay ReLU (DRLU)

The delay ReLU (DRLU)²¹ is a function that also adds a horizontal shift to the ReLU [290]; however, the DRLU uses a fixed, predetermined shift whereas RT-ReLU uses stochastic shifts (see section 3.6.11) and ABReLU computes the shift as the average of input activation map (see section 3.6.42). The DRLU is defined as

$$f(z) = \begin{cases} z - a, & z - a \geq 0, \\ 0, & z - a < 0, \end{cases} \quad (146)$$

where a is a fixed, predetermined parameter [290]. Shan, Li, and Chen also add a constraint $a > 0$ [290] and they used $a \in \{0.06, 0.08, 0.10\}$ in their experiments [290].

3.6.44 Displaced ReLU (DisReLU)

Very similar to the flexible ReLU (FReLU) (see section 4.2.15) and dynamic ReLU (DReLU) (see section 4.2.14) is the displaced ReLU (DisReLU)²² as it also shifts the ReLU [291]:

$$f(z) = \begin{cases} z, & z + a \geq 0, \\ -a, & z + a < 0, \end{cases} \quad (147)$$

where a is a predefined hyperparameter [1, 291]. A Shifted ReLU (see section 3.6.1) is a special case of DisReLU with $a = 1$ [291]. The VGG-19 [292] with DisReLUs outperform the ReLU, LReLU, PReLU, and ELU activation functions with a statistically significant difference in performance on the CIFAR-10 and CIFAR-100 datasets [158] as shown in [291].

3.6.45 Modified LReLU

Inspired by the DisReLU [291], Yang et al. proposed the modified LReLU (MLReLU) in [293]. The MLReLU is a translated LReLU and is defined as

$$f(z) = \begin{cases} z, & z + a > 0, \\ -az, & z + a \leq 0, \end{cases} \quad (148)$$

where a is a fixed parameter controlling both the slope and the threshold [293].

²¹Authors termed the function DRLU; however, the usual notation in this work would be DReLU. Since such notation would collide with the dynamic ReLU, we will use the original notation from [290] despite the inconsistency.

²²Macêdo et al. originally abbreviated the displaced ReLU as DReLU but that is already taken by dynamic ReLU from section 4.2.14.

3.6.46 Flatted-T swish

An activation function flatted-T swish (FTS) [294] combines ReLU and the logistic sigmoid activation function; it is defined as

$$f(z) = \text{ReLU}(z) \cdot \sigma(z) + T = \begin{cases} \frac{z}{1+\exp(-z)} + T, & z \geq 0, \\ T, & z < 0, \end{cases} \quad (149)$$

where T is a predefined hyperparameter [294], the recommended value is $T = -0.20$ [294]. The FTS is identical to a shifted swish for the positive z . The FTS was shown to outperform ReLU, LReLU, swish, ELU, and FReLU activation functions [294]. The special case with $T = 0$ was proposed independently under the name of ReLU-Swish in [154].

3.6.47 Optimal activation function (OAF)

The so-called Optimal Activation Function (OAF) is a combination of ReLU and swish activations proposed in [295]. It is defined as

$$f(z) = \text{ReLU}(z) + z \cdot \sigma(z) = \begin{cases} z + z \cdot \sigma(z), & z \geq 0, \\ z \cdot \sigma(z), & z < 0. \end{cases} \quad (150)$$

3.6.48 Exponential linear unit (ELU)

An ELU is an extension of LReLU where the function employs an exponential function for the negative inputs, which speeds up the learning process [236]:

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{\exp(z)-1}{a}, & z < 0, \end{cases} \quad (151)$$

where a is a hyperparameter; the authors Clevert, Unterthiner, and Hochreiter used $a = 1$ in their work [236]. The a determines the value to which an ELU saturates for inputs going to negative infinity [236].

3.6.49 Rectified exponential unit (REU)

A rectified exponential unit (REU) [296] is an activation function inspired by the ELU and swish (see sections 3.6.48 and 4.4.1) and is based on the assumption that the success of the swish activation functions is due to the non-monotonic property in the negative quadrant [296]. The REU is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ z \cdot \exp(z), & z < 0. \end{cases} \quad (152)$$

A parametric version called parametric rectified exponential unit (PREU) was also proposed in [296]; see section 4.2.9 for details.

3.6.50 Apical dendrite activation (ADA)

A biologically inspired AF named apical dendrite activation (ADA) was proposed in [297]. It is similar to the ELU, but it applies an exponential function for positive inputs. It is defined as

$$f(z) = \begin{cases} \exp(-az + b), & z \geq 0, \\ 0, & z < 0, \end{cases} \quad (153)$$

where a and b are fixed parameters [297].

3.6.51 Leaky apical dendrite activation (LADA)

As LReLU extends the ReLU, the leaky apical dendrite activation (LADA) [297] extends the ADA.

$$f(z) = \begin{cases} \exp(-az + b), & z \geq 0, \\ cz, & z < 0, \end{cases} \quad (154)$$

where a , b , and $c \in [0, 1]$ are fixed parameters [297]. Georgescu et al. used $c = 0.01$ in their experiments in [297].

3.6.52 Sigmoid linear unit (SigLU)

The sigmoid linear unit (SigLU)²³ is an ELU alternative that uses a modified logistic sigmoid instead of the exponential [170]. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{1 - \exp(-2z)}{1 + \exp(-2z)}, & z < 0. \end{cases} \quad (155)$$

3.6.53 Swish and ReLU activation (SaRa)

The swish and ReLU activation (SaRa) is an AF combining the swish and ReLU AFs proposed in [298]. It is defined²⁴ as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{z}{1 + a \cdot \exp(-bz)}, & z < 0, \end{cases} \quad (156)$$

where a and b are fixed parameters; Qureshi and Sarosh Umar recommend $a = 0.5$ and $b = 0.7$ [298].

3.7 Maxsig

The maxsig is one of the AFs listed in [96]. The maxsig is similar to the SigLU (see section 3.6.52) and is defined as

$$f(z) = \max(z, \sigma(z)), \quad (157)$$

where $\sigma(z)$ is the logistic sigmoid [96].

3.7.1 Tanh linear unit (ThLU)

The tanh linear unit (ThLU) [299]²⁵ is an AF combining tanh and ReLU. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{2}{1 + \exp(-z)} - 1, & z < 0, \end{cases} = \begin{cases} z, & z \geq 0, \\ \tanh\left(\frac{z}{2}\right), & z < 0. \end{cases} \quad (158)$$

The ThLU is a special case of the tanh based ReLU (TReLU) with $b_i = \frac{1}{2}$. Similar AF was used under the name maxtanh in [96] — it just omitted the scaling factor. The maxtanh can also be written as $f(z) = \max(z, \tanh(z))$ [96].

3.7.2 DualELU

The DualELU [279] is equivalent of DualReLU (see section 3.6.36) for ELUs and are defined as

$$f(z, z') = f_{\text{EL}}(z) - f_{\text{EL}}(z'), \quad (159)$$

where $f_{\text{EL}}(z)$ is the ELU activation function applied to an input z .

3.7.3 Difference ELU (DiffELU)

An ELU variant named difference exponential linear unit (DiffELU)²⁶ was proposed in [300]. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ a(z \exp(z) - b \exp(bz)), & z < 0, \end{cases} \quad (160)$$

where a and $b \in (0, 1)$ are fixed parameters [300]. Hu et al. also tested setting the parameters to be trainable but that led to worse performance [300]. The recommended setting is $a = 0.3$ and $b = 0.1$ [300].

²³The AF is unnamed in the original work [170].

²⁴The formula in [298] is malformed; we believe that this is the intended case. It is possible that authors intended that the SaRa is actually only the part that is defined for the negative inputs in eq. (156) — however, we think that it is less likely as that would be only a swish (see section 4.4.1) AF with some fixed scaling of the output or the AHAF (see section 4.4.2) with fixed parameters.

²⁵The ref [299] is not the original work with ThLUs; it references another work but that uses pure tanh as the AFs.

²⁶Hu et al. used the abbreviation DELU but this name is used for the AF proposed by Pishchik in [252] throughout this work.

3.7.4 Polynomial linear unit (PolyLU)

The polynomial linear unit (PolyLU) is an AF similar to the ELU proposed in [301]. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{1}{1-z} - 1, & z < 0. \end{cases} \quad (161)$$

Despite the similarity with the ELU, Feng and Yang have shown that the PolyLU outperformed the ELU on the CIFAR-10/100 [158] and Dogs vs. Cats [302, 303] datasets [301]. The PolyLU was also proposed under the name first power linear unit with sign (FPLUS)²⁷ in [304].

3.7.5 Inverse polynomial linear unit (IpLU)

The polynomial linear unit (IpLU) was proposed in [269]; it is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ \frac{1}{1+|z|^a}, & z < 0, \end{cases} \quad (162)$$

where $a > 0$ is a fixed hyperparameter guaranteeing a small slope for negative inputs [269].

3.7.6 Power linear unit (PoLU)

The power linear unit (PoLU) [305] is an AF similar to the ELU. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ (1-z)^{-a} - 1, & z < 0, \end{cases} \quad (163)$$

where a is a fixed parameter [305]. Li, Ding, and Li used $a \in \{1, 1.5, 2\}$ in their experiments in [305].

3.7.7 Power function linear unit (PFLU)

The power function linear unit (PFLU) is an AF proposed in [306]; it is defined as

$$f(z) = z \cdot \frac{1}{2} \left(1 + \frac{z}{\sqrt{1+z^2}} \right). \quad (164)$$

3.7.8 Faster power function linear unit (FPFLU)

The faster power function linear unit (FPFLU) is an AF proposed in [306] that resembles the IpLU (see section 3.7.5) It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ z + \frac{z^2}{\sqrt{1+z^2}}, & z < 0. \end{cases} \quad (165)$$

3.7.9 Elastic adaptively parametric compounded unit (EACU)

The elastic adaptively parametric compounded unit (EACU) [307] is a stochastic AF. It is defined as

$$f(z_i) = \begin{cases} b_i z_i, & z_i \geq 0, \\ a_i z_i \cdot \tanh(\ln(1 + \exp(a_i, z_i))), & z_i < 0, \end{cases} \quad (166)$$

where b_i is stochastically sampled during training as

$$b_i = \begin{cases} s_i, & 0.5 < s_i < 1.5, \\ 1, & \text{otherwise,} \end{cases} \quad (167)$$

$$s_i \sim \text{mathrm{N}}(0, 0.01), \quad (168)$$

and a_i is an adaptive parameter for each neuron or channel i [307].

²⁷Duan, Yang, and Dai used the equivalent definition $f(z) = (\text{sgn}(z) \cdot z + 1)^{\text{sgn}(z)} - 1$ in [304], hence the name.

3.7.10 Lipschitz ReLU (L-ReLU)

A L-ReLU [308] is a piecewise linear activation function. The slope of the negative part is selected with respect to a data-dependent Lipschitz constant [308]. It builds on a proposed piecewise function that treats the positive $z > 0$ and negative values ($z \leq 0$) separately:

$$f(z) = p(z|z > 0) + n(z|z \leq 0), \quad (169)$$

where

$$p(z) = \max(\phi(z), 0), \quad (170)$$

and

$$n(z) = \min(\mu(z), 0), \quad (171)$$

where $\phi(z)$ and $\mu(z)$ can be any function $f : \mathbb{R} \rightarrow \mathbb{R}$ [308]. This makes the positive part of the piecewise lay in the first quadrant of the Cartesian coordinate system and the negative part in the third quadrant [308].

3.7.11 Scaled exponential linear unit (SELU)

A SELU [309] was proposed in order to make the network self-normalize by automatically converging towards zero mean and unit variance [1]. The ELU was chosen as the basis for self-normalizing neural networks (SNNs) because these cannot be derived with ReLUs, sigmoid, and tanh units or even LReLUs [309] — the activation function has to have negative and positive values for controlling the mean, saturation region where derivatives approach zero in order to dampen the variance if it is too large, a slope larger than one in order to increase the variance if it is too small, and a continuous curve to ensure a fixed point where the variance dampening is balanced out by the variance increasing [309]. The SELU is defined as

$$f(z) = \begin{cases} az, & z \geq 0, \\ ab(\exp(z) - 1), & z < 0, \end{cases} \quad (172)$$

where $a > 1$ and b are predefined parameters [1, 309]; the recommended values are $a \approx 1.05078$ and $b \approx 1.6733$ [309].

3.7.12 Leaky scaled exponential linear unit (LSELU)

A leaky variant of SELU called leaky scaled exponential linear unit (LSELU) was proposed in [310] and is defined as

$$f(z) = \begin{cases} az, & z \geq 0, \\ ab(\exp(z) - 1) + acz, & z < 0, \end{cases} \quad (173)$$

where $a > 1$ and b are predefined parameters of the original SELU (see section 3.7.11), and c is a new, predefined parameter controlling the leakiness of the unit [310].

3.7.13 Scaled exponentially-regularized linear unit (SERLU)

The scaled exponentially-regularized linear unit (SERLU) is a modification of the SELU proposed in [311]; it is defined as

$$f(z) = \begin{cases} az, & z \geq 0, \\ abz \exp(z), & z < 0, \end{cases} \quad (174)$$

where $a > 0$ and $b > 0$ are predefined parameters [311]. An extension of this approach named ASERLU for bidirectional long short-term memory (BiLSTM) architectures was proposed in [312].

3.7.14 Scaled scaled exponential linear unit (sSELU)

Additional scaling of the negative pre-activations was introduced in the scaled scaled exponential linear unit (sSELU) [310]:

$$f(z) = \begin{cases} az, & z \geq 0, \\ ab(\exp(cz) - 1), & z < 0, \end{cases} \quad (175)$$

where $a > 1$ and b are predefined parameters of the original SELU (see section 3.7.11), and c is a new, predefined parameter controlling the scaling of the negative inputs to the unit [310].

3.7.15 RSigELU

A parametric ELU variant called RSigELU [313] is defined as

$$f(z) = \begin{cases} z \left(\frac{1}{1+\exp(-z)} \right) a + z, & 1 < z < \infty, \\ z, & 0 \leq z \leq 1, \\ a(\exp(z) - 1), & -\infty < z < 0, \end{cases} \quad (176)$$

where a is a predefined parameter, Kiliçarslan and Celik used $0 < a < 1$ in their work [313]. For $a = 0$, the RSigELU becomes ReLU [313]. The RSigELU was shown to outperform ReLU, LReLU, softsign, swish, ELU, SEU, GELU, LISA, Hexpo and softplus on the MNIST dataset [182], Fashion MNIST [314] and the IMDB Movie dataset; it still outperformed these activation functions on the CIFAR-10 dataset [158] but it was outperformed by its variant RSigELUD [313].

3.7.16 HardSReLU

Another AF proposed by Kiliçarslan is the HardSReLU [315]. Kiliçarslan defined the AF as

$$f(z) = \begin{cases} az \left(\max \left(0, \min \left(1, \frac{z+1}{2} \right) \right) \right) + z, & z \geq 0, \\ a(\exp(z) - 1), & z < 0, \end{cases} \quad (177)$$

where a is a fixed slope parameter [315].

3.7.17 Exponential linear sigmoid squashing (ELiSH)

An activation function exponential linear sigmoid squashing (ELiSH) [101] combines the swish (see section 4.4.1) and the ELU function [1]. It is defined as

$$f(z) = \begin{cases} \frac{z}{1+\exp(-z)}, & z \geq 0, \\ \frac{\exp(z)-1}{1+\exp(-z)}, & z < 0. \end{cases} \quad (178)$$

3.7.18 Hard exponential linear sigmoid squashing (HardELiSH)

As ELiSH (see section 3.7.17) combines swish with ELU and linear function, the hard exponential linear sigmoid squashing (HardELiSH) combines the Hard sigmoid [253] with ELU and linear function [101]. It is defined as

$$f(z) = \begin{cases} z \cdot \max \left(0, \min \left(\frac{z+1}{2}, 1 \right) \right), & z \geq 0, \\ (1 + \exp(-z)) \cdot \max \left(0, \min \left(\frac{z+1}{2}, 1 \right) \right), & z < 0. \end{cases} \quad (179)$$

3.7.19 RSigELUD

The RSigELUD is a double parameter variant of the RSigELU (see section 3.7.15) [313] that is defined as

$$f(z) = \begin{cases} z \left(\frac{1}{1+\exp(-z)} \right) a + z, & 1 < z < \infty, \\ z, & 0 \leq z \leq 1, \\ b(\exp(z) - 1), & -\infty < z < 0, \end{cases} \quad (180)$$

where a and b are predefined parameters, Kiliçarslan and Celik used $0 < a < 1$ and $0 < b < 1$ in their work [313]. For $a = b = 0$, the RSigELUD becomes the ReLU the same as the RSigELU; however, for $a = 0$ and positive b , the function resembles the vanilla ELU [313].

3.7.20 LS-ReLU

The LS-ReLU²⁸ is a ReLU-inspired AF proposed in [316]. It is defined as

$$f(z) = \begin{cases} \frac{z}{1+|z|}, & z \leq 0, \\ z, & 0 \leq z \leq b, \\ \log(az + 1) + |\log(ab + 1) - b|, & z \geq b, \end{cases} \quad (181)$$

where a and b are fixed²⁹ parameters [316].

²⁸Not an abbreviation.

²⁹Wang et al. do not specify whether the parameters are trainable or fixed.

3.8 Square-based activation functions

Several square-based activation functions were proposed in [317–319] for better computational efficiency, especially on low-power devices [317]. The approach uses the square function to replace the potentially costly exponential function. These function leads to significantly more efficient computation when there is no hardware implementation of the exponential function [317]. The efficiency gains can be further improved with a custom hardware operator

$$f_h(x) = -|x| \cdot x, \quad (182)$$

which can be used for efficient hardware implementation of all of the activation functions of the square-based family [317]. The usage of the AFs from the family can lead to performance gains of one order of magnitude compared to traditional AFs [317] for both forward and backward passes (depends on the particular activation function and the usage of fixed or floating point representations) [317].

3.8.1 SQNL

A computationally efficient activation function was proposed in [318]; unlike many other sigmoidal functions, it uses the square operator instead of the exponential function in order to achieve better computational efficiency. The derivative of the function is linear, which leads to a less computationally costly computation of the gradient. The function is defined in [317] (the original paper [318] had several mistakes in the definition) as

$$f(z) = \begin{cases} 1, & z > 2, \\ z - \frac{z^2}{4}, & 0 \leq z \leq 2, \\ z + \frac{z^2}{4}, & -2 \leq z < 0, \\ -1, & z < -2. \end{cases} \quad (183)$$

The SQNL³⁰ has bounded range $[-1, 1]$ [317]. The performance of the SQNL was verified on several datasets from the UCI Machine Learning Repository [320] and on the MNIST dataset [182]; more details available in [318].

3.8.2 Square linear unit (SQLU)

Similarly as the SQNL (see section 3.8.1) uses square function to form a sigmoidal function to approximate tanh, the square linear unit (SQLU) [317] uses square function to form a ELU-like activation function that is computationally efficient:

$$f(z) = \begin{cases} z, & z > 0, \\ z + \frac{z^2}{4}, & -2 \leq z \leq 0, \\ -1, & z < -2. \end{cases} \quad (184)$$

The SQLU basically uses the negative part of the SQNL and replaces the positive part with a linear function.

3.8.3 Square swish (squish)

Another example of the family of activation functions based on the square operator is the square swish (squish) [317], which is an AF inspired by the swish and GELU (see section 3.3.1). It uses the square non-linearity in order to achieve good computational efficiency:

$$f(z) = \begin{cases} z + \frac{z^2}{32}, & z > 0, \\ z + \frac{z^2}{2}, & -2 \leq z \leq 0, \\ 0, & z < -2. \end{cases} \quad (185)$$

While the squish was inspired by the swish and GELU activation functions, it is an approximation of neither [317].

3.8.4 Square REU (SqREU)

Similarly as REU (see section 3.6.49) is a combination of the ReLU and swish activation functions, the *square REU* (SqREU) [317] is a combination of ReLU and squish:

$$f(z) = \begin{cases} z, & z > 0, \\ z + \frac{z^2}{2}, & -2 \leq z \leq 0, \\ 0, & z < -2. \end{cases} \quad (186)$$

³⁰SQNL is not an abbreviation but rather a name given by Wuraola and Patel.

3.8.5 Square softplus (SqSoftplus)

A square softplus (SqSoftplus) is another square-based computationally efficient replacement of an activation function — softplus [317]:

$$f(z) = \begin{cases} z, & z > \frac{1}{2}, \\ z + \frac{(z + \frac{1}{2})^2}{2}, & -\frac{1}{2} \leq z \leq \frac{1}{2}, \\ 0, & z < -\frac{1}{2}. \end{cases} \quad (187)$$

3.8.6 Square logistic sigmoid (LogSQL)

While the SQL [318] replaces the tanh AF, the square logistic sigmoid (LogSQL) [317] is a square-based replacement for the logistic sigmoid:

$$f(z) = \begin{cases} 1, & z > 2, \\ \frac{1}{2} \left(z - \frac{z^2}{4} \right) + \frac{1}{2}, & 0 \leq z \leq 2, \\ \frac{1}{2} \left(z + \frac{z^2}{4} \right) + \frac{1}{2}, & -2 \leq z < 0, \\ 0, & z < -2. \end{cases} \quad (188)$$

3.8.7 Square softmax (SQMAX)

The square softmax (SQMAX) is a square-based replacement for the softmax, which is exponential-based. It is defined as

$$f(z_j) = \frac{(z_j + c)^2}{\sum_{k=1}^N (z_k + c)^2}, \quad (189)$$

where $f(z_j)$ is the output of a neuron j in a softmax layer consisting of N neurons and $c = 4$ is a predefined constant [317].

3.8.8 Linear quadratic activation

Another square-based AF called linear quadratic activation (LinQ) was proposed in [319].

$$f(z) = \begin{cases} az + (1 - 2z + z^2), & z \geq 2 - 2a, \\ \frac{1}{4}z(4 - |z|), & -2 + 2a < z < 2 - 2a, \\ az - (1 - 2z + z^2), & z \leq -2 + 2a, \end{cases} \quad (190)$$

where a is a fixed parameter controlling the slope of the function's linear parts [319].

3.8.9 Inverse square root linear unit (ISRLU)

Inverse square root linear unit (ISRLU) [321] is an activation function similar to the ELU (see section 3.6.48). It has similar properties and a shape as ELU; however, it is faster to compute, leading to more efficient training and inference [321]. It is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ z \cdot \frac{1}{\sqrt{1 + az^2}}, & z < 0, \end{cases} \quad (191)$$

where a is a hyperparameter controlling the value to which the ISRLU saturates for negative inputs [321]. While the authors state that the hyperparameter a could be trainable for each neuron i , only the non-trainable variant was analyzed [321]. Carlile et al. analysed ISRLU with $a = 1$ and $a = 3$ [321].

3.8.10 Inverse square root unit (ISRU)

ISRU [321] is an activation function meant to replace sigmoidal activation functions. It is defined as

$$f(z) = z \cdot \frac{1}{\sqrt{1 + az^2}}, \quad (192)$$

where a is a fixed hyperparameter controlling the saturation values; the parameter could be trainable similarly as in the ISRLU (see section 3.8.9) but only the nonadaptive variant was used [321].

3.8.11 Modified Elliott function (MEF)

The modified Elliott function (MEF) [164] is an AF inspired by the Elliott function (see section 3.2.15); it can also be considered to be a translated special case of the ISRU (see section 3.8.10) with $a = 1$. It is defined as

$$f(z) = z \cdot \frac{1}{\sqrt{1+z^2}} + \frac{1}{2}. \quad (193)$$

3.9 Square-root-based activation function (SQRT)

A square-root-based activation function (SQRT) is a monotonically increasing, unbounded activation function proposed in [322] with a similar structure as the earlier proposed logarithmic activation function (LAF) but with the square root function instead of the natural logarithm used in LAF. It is defined as

$$f(z) = \begin{cases} \sqrt{z}, & z \geq 0, \\ -\sqrt{-z}, & z < 0. \end{cases} \quad (194)$$

The SQRT activation function was found to outperform both tanh and ReLU activation functions on the CIFAR-10 dataset [158] in experiments in [322].

A parametric variant of the SQRT called S-shaped activation function (SSAF) was proposed independently in [323]. It is defined as

$$f(z) = \begin{cases} \sqrt{2az}, & z \geq 0, \\ -\sqrt{-2az}, & z < 0, \end{cases} \quad (195)$$

where a is a fixed parameter [323].

3.10 Bent identity

The bent identity [324] is an AF approximating the ReLU; it can be seen as a fixed variant of the bendable linear unit (BLU) (see section 4.2.37) with $a_i = \frac{1}{2}$. It is defined as

$$f(z) = \frac{\sqrt{z^2 + 1} - 1}{2} + z. \quad (196)$$

3.11 Mishra activation function

The Mishra³¹ AF is defined as

$$f(z) = \frac{1}{2} \left(\frac{z}{1+|z|} \right)^2 + \frac{1}{2} \frac{z}{1+|z|}. \quad (197)$$

3.12 Saha-Bora activation function (SBAF)

A Saha-Bora activation function (SBAF) was proposed in [286, 326] to be used for the habitability classification of exoplanets. It employs two non-trainable parameters α and k , which were set to $k = 0.98$ and $\alpha = 0.5$, where authors determined a stable fixed point. It is defined as:

$$f(z) = \frac{1}{1 + kz^\alpha(1-z)^{(1-\alpha)}}. \quad (198)$$

3.13 Logarithmic activation function

The logarithmic activation function (LAF) was proposed in [327] (ref. from [152]). According to [152], it is defined as

$$f(z) = \begin{cases} \ln z + 1, & z \geq 0, \\ -\ln -z + 1, & z < 0. \end{cases} \quad (199)$$

The LAF was independently proposed under the name symlog in [328].

³¹The AF was unnamed in the original papers [73, 325]; however, the work [71] named it using the name of the original author. We keep the naming in this work.

3.14 Symexp

The symexp [328] is an activation function that is inverse of the logmoid activation unit (LAU). It is defined as

$$f(z) = \text{sgn}(z) (\exp(|z|) - 1). \quad (200)$$

3.15 Scaled polynomial constant unit (SPOCU)

The scaled polynomial constant unit (SPOCU) is a polynomial-based AF proposed in [329, 330]. It is defined as

$$f(z) = ah \left(\frac{z}{c} + b \right) - ah(b), \quad (201)$$

where

$$h(x) = \begin{cases} r(d), & x \geq d, \\ r(x), & 0 \leq x < d, \\ x, & x < 0, \end{cases} \quad (202)$$

$$r(x) = x^3 (x^5 - 2x^4 + 2), \quad (203)$$

and $a > 0$, $b \in (0, 1)$, $c > 0$, and $d \in [1, \infty)$ are fixed parameters satisfying additional conditions listed in [329, 330].

3.16 Polynomial universal activation function (PUAF)

Similarly as the universal activation function (UAF) (see section 4.22), the polynomial (PUAF)³² is able to approximate popular AFs such as the logistic sigmoid, ReLU, and swish [331]. It is defined as

$$f(z) = \begin{cases} z^a, & z > c, \\ z^a \frac{(c+z)^b}{(c+z)^b + (c-z)^b}, & |z| \leq c, \\ 0, & z < -c, \end{cases} \quad (204)$$

where a , b and c are fixed parameters [331]. The PUAF becomes the ReLU with $a = 1$, $b = 0$, and $c = 0$; the logistic sigmoid is approximated with $a = 0$, $b = 5$, and $c = 10$; finally, the swish is approximated using $a = 1$, $b = 5$, and $c = 10$ [331].

3.17 Softplus

The softplus function was proposed in [333] and is defined as

$$f(z) = \ln(\exp(z) + 1). \quad (205)$$

The softplus was used as an activation function in [222] where it was used alongside with a ReLU. The advantage of softplus over ReLU is that it is smooth and it has a non-zero gradient for negative inputs; thus, it does not suffer from the phenomenon of dying out neurons that is common in networks with ReLU activations [334]. The softplus was found to outperform ReLU for certain applications and architectures [334]. A noisy variant was used for spiking neural networks in [335].

3.18 Parametric softplus (PSoftplus)

Parametric softplus (PSoftplus) [336] is a softplus variant that allows for scaling and shifting using two additional parameters. The PSoftplus is defined as

$$f(z) = a (\ln(\exp(z) + 1) - b), \quad (206)$$

where a and b are fixed predetermined hyperparameters [336]. The creation of the softplus was motivated by the assumption that activations with mean outputs close to zero can improve the performance of a neural network; since the output of the softplus is always positive, a shift parameter b was introduced to shift the mean output closer to zero [336]. The slope controlling parameter a is used to adjust the function and the gradient disappearance or overflow during training [336]. The recommended values are $a = 1.5$ and $b = \ln(2)$ [336].

³²Hwang and Kim named the function only as the *universal activation function* but this name is already taken by the UAF by Yuen et al. from [332].

3.18.1 Soft++

Another softplus extension *Soft++* is a multiparametric nonsaturating nonlinear activation function proposed in [337]. It is defined as

$$f(z) = \ln(1 + \exp(az)) + \frac{z}{b} - \ln(2), \quad (207)$$

where a and b are fixed predetermined hyperparameters [337]; however, Ciuparu, Nagy-Dăbâcan, and Mureşan proposed they could be adaptable in future works. Multiple values of the parameters were used in the experiments in [337], but $a = 1$ and $b = 2$ were found to work well [337]; nevertheless, a hyperparameter optimization is recommended [337].

3.19 Rand softplus (RSP)

A softplus variant rand softplus (RSP) [338] introduces a stochastic parameter a_l that is determined by the noise level of the input data [338]. The RSP is defined as

$$f(z_l) = (1 - a_l) \max(0, z_l) + a_l \cdot \ln(1 + \exp(z_l)), \quad (208)$$

where a_l is adapting to the input noise levels of each layer l — the exact procedure is described in [338].

3.20 Aranda-Ordaz

The Aranda-Ordaz AF[339, 340] was used in NNs in [339]. It is defined as

$$f(z) = 1 - (1 + a \exp(z))^{-\frac{1}{a}}, \quad (209)$$

where $a > 0$ is a fixed parameter [339]. Essai Ali, Abdel-Raman, and Badry used $a = 2$ in their work [74].

3.21 Bi-firing activation function (bfire)

A bi-firing activation function (bfire) was proposed in [341] and is defined as

$$f(z) = \begin{cases} z - \frac{a}{2}, & z > a, \\ \frac{z^2}{2a}, & -a \geq z \geq a \\ -z - \frac{a}{2}, & z < -a, \end{cases} \quad (210)$$

where a is a predefined smoothing hyperparameter [341]. The bfire is basically a smoothed variant of the later proposed vReLU (see section 3.6.25) as it becomes vReLU as $a \rightarrow 0$.

3.22 Bounded bi-firing activation function (bbfire)

A bounded variant of the bi-firing (bfire) activation function (see section 3.21) called bbfire was proposed in [251]; similarly as BReLU and BLReLU bounds ReLU and LReLU respectively (see sections 3.6.16 and 3.6.24), the bounded bi-firing function (bbfire) is defined as

$$f(z) = \begin{cases} b, & z < -b - \frac{a}{2}, \\ -z - \frac{a}{2}, & -b - \frac{a}{2} \geq z < -a, \\ \frac{z^2}{2a}, & -a \geq z \geq a, \\ z - \frac{a}{2}, & a < z \leq b + \frac{a}{2}, \\ b, & z > b + \frac{a}{2}, \end{cases} \quad (211)$$

where a and b are predefined hyperparameters [251]. The is symmetrical about the origin and has a near inverse-bell-shaped activation curve [251]. While authors of the original bfire [341] solved potential numerical instabilities caused by the unboundedness by imposing a small L_1 penalty on the hidden activation values [341], the bbfire alleviates this problem explicitly without any need for such penalty.

3.23 Piecewise Mexican-hat activation function (PMAF)

The piecewise Mexican-hat activation function (PMAF) was used in [342]; it is defined as

$$f(z) = \begin{cases} \left(\frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}\right) \left(1 - (z+a)^2\right) \exp\left(-\frac{(z+a)^2}{2}\right), & z < 0, \\ \left(\frac{2}{\sqrt{3}}\pi^{-\frac{1}{4}}\right) \left(1 - (z-a)^2\right) \exp\left(-\frac{(z-a)^2}{2}\right), & z \geq 0, \end{cases} \quad (212)$$

where a is a fixed parameter — Liu, Zeng, and Wang used $a = 4$ [342].

3.24 Piecewise radial basis function (PRBF)

The piecewise (PRBF) was used in [342]; it is defined as

$$f(z) = \begin{cases} \exp\left(-\frac{(z-2a)^2}{b^2}\right), & z \geq a \\ \exp\left(-\frac{z^2}{b^2}\right), & -a < z < a \\ \exp\left(-\frac{(z+2a)^2}{b^2}\right), & z \leq -a \end{cases} \quad (213)$$

where a and b are fixed parameters [342] — Liu, Zeng, and Wang used $a = 3$ and $b = 1$ [342].

3.25 Comb-H-sine

A comb-H-sine is an activation function that was found using an evolutionary approach in [109]. It is defined as

$$f(z) = \sinh(az) + \sinh^{-1}(az), \quad (214)$$

where $\sinh(x)$ is the hyperbolic sine, $\sinh^{-1}(x)$ is its inverse, and a is a predefined hyperparameter [109]. This function was found to outperform ReLU, tanh, logistic sigmoid, and several other activation functions in LSTM models in [109].

3.26 Modified arcsinh

The modified arcsinh (m-arcsinh) AF was proposed in [343] and is defined as

$$f(z) = \frac{1}{12} \sinh^{-1}(z) \sqrt{|z|}. \quad (215)$$

Interestingly, the m-arcsinh can be used either as an AF in a NN or as a kernel function in the support vector machine (SVM) [343].

3.27 hyper-sinh

The hyper-sinh is an AF that uses the sinh and cubic functions [344, 345]; it is defined as

$$f(z) = \begin{cases} \frac{\sinh(z)}{3}, & z > 0, \\ \frac{z^3}{4}, & z \leq 0. \end{cases} \quad (216)$$

3.28 Arctid

The arctid is an arctan-based AF used in [96]; it is defined as

$$f(z) = (\tan^{-1}(z))^2 - z. \quad (217)$$

3.29 Sine

The sine with inputs scaled by π was used as an activation in [346]:

$$f(z) = \sin(\pi z). \quad (218)$$

It was, for example, used recently with a data-driven determination of a network's biases in [139]. Just the sine function without any scaling was used as an activation in [347–352].

Scaled sine with vertical shift was used in [117]; the used AF is defined as

$$f(z) = 0.5 \sin(az) + 0.3, \quad (219)$$

where a is a fixed parameter; $a \in \{0.2, 0.8, 1.2, 1.8, 4\}$ [117].

3.30 Cosine

A cosine activation was used in simulations in [353]; it was defined as

$$f(z) = 1 - \cos(z). \quad (220)$$

3.31 Cosid

The cosid is one of the AFs listed in [96]. It is defined as

$$f(z) = \cos(z) - z. \quad (221)$$

3.32 Sinp

A parametric AF similar to the cosid was proposed in [354] under the name sinp.³³ It is defined as

$$f(z) = \sin(z) - az, \quad (222)$$

where a is a fixed parameter [354]. Chan et al. used $a \in \{1, 1.5, 2\}$ [354].

3.33 Growing cosine unit (GCU)

Another cosine-based AF is the growing cosine unit (GCU) proposed in [355]. It is defined as

$$f(z) = z \cos(z). \quad (223)$$

Empirical evaluation of the performance of GCU compared to ReLU, PReLU, and mish is available in [356]; its brief evaluation with respect to the generation of NFTs is available in [357].

3.34 Amplifying sine unit (ASU)

The amplifying sine unit (ASU) is the sine equivalent of the GCU [358, 359]

$$f(z) = z \sin(z). \quad (224)$$

3.35 Sinc

The sinc is an older AF proposed in [360]. It is defined as

$$f(z) = \begin{cases} \frac{\sin(\pi z)}{\pi z}, & z \neq 0, \\ 1, & z = 0. \end{cases} \quad (225)$$

A shifted variant was proposed under the name shifted sine unit (SSU) in [361]. It is defined as

$$f(z) = \pi \text{sinc}(z - \pi). \quad (226)$$

3.36 Decaying sine unit (DSU)

The decaying sine unit (DSU) is a sinc based AF proposed in [361]. It is defined as

$$f(z) = \frac{\pi}{2} (\text{sinc}(z - \pi) - \text{sinc}(z + \pi)). \quad (227)$$

3.37 Hyperbolic cosine linearized squashing function (HcLSH)

The hyperbolic cosine linearized squashing function (HcLSH) is an AF proposed in [362]; it is defined as

$$f(z) = \begin{cases} \ln(\cosh(z) + z \cdot \cosh(\frac{z}{2})), & z \geq 0, \\ \ln(\cosh(z)) + z, & z < 0. \end{cases} \quad (228)$$

3.38 Polyexp

The polyexp is an AF combining quadratic function and an exponential function [360];³⁴ it is defined as

$$f(z) = (az^2 + bz + c) \exp(-dz^2), \quad (229)$$

where a, b, c , and d are fixed parameters [360].

³³Technically, the full name used by Chan et al. is SinP[N] but we omitted the parameter from the name of the AF.

³⁴The [346] is referenced as the origin of polyexp in [360] but we have not seen the definition there.

3.39 Exponential

The exponential was used as an AF in [117]. The AF was defined as

$$f(z) = \exp(-z). \quad (230)$$

3.40 E-Tanh

An AF named E-Tanh combining the exponential and tanh functions was proposed in [363]. It is defined as

$$f(z) = a \cdot \exp(z) \tanh(z), \quad (231)$$

where a is a fixed scaling parameter [363, 364].

3.40.1 Evolved combination of tanh and ReLU

The combination of tanh and ReLU was found using neuroevolution in [116] — while Vijayaprabakaran and Sathiyamurthy also mentioned other AFs, this combination led to the best performance on the HAR dataset using the long short-term memory (LSTM) units. The best-performing recurrent AF was

$$f(z) = a (\tanh(z^2) + \text{ReLU}(z)) \quad (232)$$

and the regular AF was

$$f(z) = \max(\tanh(\log(z)), \text{ReLU}(z)). \quad (233)$$

See [116] for evaluation details and for other top AFs.

3.41 Wave

The wave is an AF combining quadratic function and an exponential function [360];³⁵ similarly as the polyexp but only with a single parameter; it is defined as

$$f(z) = (1 - z^2) \exp(-az^2), \quad (234)$$

where a is a fixed parameter [360].

3.42 Non-monotonic cubic unit (NCU)

A simple AF based on a third-degree polynomial was proposed in [361]. It is named non-monotonic cubic unit (NCU) and is defined as

$$f(z) = z - z^3. \quad (235)$$

3.43 Triple

Another AF based on a third-degree polynomial called triple was proposed in [365]. It is defined as

$$f(z) = a \cdot z^3, \quad (236)$$

where a is a fixed parameter [365]. Chen et al. tested values of $a \in \{0.1, 0.5, 1, 2\}$ and observed that $a = 1$ reaches the best results [365].

3.44 Shifted quadratic unit (SQU)

The shifted quadratic unit (SQU) [361] is a simple non-monotonic AF defined as

$$f(z) = z^2 + z. \quad (237)$$

³⁵The [346] is referenced as the origin of wave in [360] but we have not seen the definition there.

3.45 Knowledge discovery activation function (KDAC)

Wang et al. proposed a special AF for knowledge discovery in [366]. This function named knowledge discovery activation function (KDAC) has two adaptive parameters $a > 0$ and $b > 0$ and one fixed parameter c . It is defined³⁶ as

$$f(z) = p \cdot (1 - h_{\max}(p, r)) + r \cdot h(p, r) + kh_{\max}(p, r)(1 - h_{\max}(p, r)), \quad (238)$$

where

$$h_{\max}(x, y) = \text{clip}\left(\frac{1}{2} - \frac{1}{2} \frac{x - y}{c}\right), \quad (239)$$

$$\text{clip}(x) = \begin{cases} 0, & x \leq 0, \\ x, & 0 < x < 1, \\ 1, & x \geq 1, \end{cases} \quad (240)$$

$$p = az, \quad (241)$$

$$q = h_{\min}(bz, s), \quad (242)$$

$$r = \begin{cases} p, & z > 0, \\ bz \cdot (1 - q) + s \cdot h(q, s) + kq(1 - q), & z \leq 0, \end{cases} \quad (243)$$

$$s = \tanh(z), \quad (244)$$

and

$$h_{\min}(x, y) = \text{clip}\left(\frac{1}{2} + \frac{1}{2} \frac{x - y}{c}\right). \quad (245)$$

Wang et al. used fixed $c = 0.01$ [366].

3.46 K-winner-takes-all activation function (k-WTA)

The k-winner-take-all (k-WTA) AF was used to improve adversarial robustness in [367]. It is defined as

$$f(z)_j = \begin{cases} z_j, & z_j \in \{k \text{ largest elements of } z\}, \\ 0, & \text{otherwise,} \end{cases} \quad (246)$$

where $f(z) : \mathbb{R}^N \rightarrow \mathbb{R}^N$ is the k-WTA AF and $f(z)_j$ its j -th element, z is the input to the AF, and k a fixed parameter [367].

3.47 Volatility-based activation function (VBAF)

The volatility-based activation function (VBAF)³⁷ is an AF with multiple inputs proposed in [368]. It is meant for time-series forecasting and was used in a LSTM NN in [368]. It is defined as

$$f(z_1, \dots, z_n) = \sqrt{\frac{\sum_{j=1}^n (\bar{0}z - z_j)^2}{n}}, \quad (247)$$

where

$$\bar{0}z = \frac{\sum_{j=1}^n z_j}{n}, \quad (248)$$

n is the number of time-series samples in the given period [368, 369]. Unfortunately, no more details about the application of the VBAF were provided in [368]; thus, it remains unclear whether the VBAF was applied only directly to the inputs, or it was used on intermediary representations of a NN.

3.48 Chaotic activation functions

The chaotic activation functions (CAFs) listed in this work are AFs that use a recursive definition to produce a chaotic behavior.

³⁶The original code by Wang et al. is available at <https://github.com/pyy-copyto/KDAC/blob/main/KDAC.py>.

³⁷Kayim and Yilmaz named the function originally only *volatility activation function*.

3.48.1 Hybrid chaotic activation function

The hybrid chaotic activation function (HCAF) is a multi-output type of AF proposed in [370]. Neuron i in layer l takes an input z_i^l , applies the logistic sigmoid AF and then maps the outputs using logistic map function to individual outputs going to the neurons in layer $l + 1$ [370]. Therefore, a single neuron in layer l emits a different activation value to each neuron in the layer $l + 1$ [370].

For a neuron i , the HCAF first applies the logistic sigmoid function to produce activation a_i

$$a_i = f(z_i) = \sigma(z_i), \quad (249)$$

then the first value going to the neuron 1 in the following layer is calculated as

$$c_{i,1} = r a_i (1 - a_i), \quad (250)$$

and the output values going to the other neurons in the following layer are calculated recursively as

$$c_{i,j} = r c_{i,j-1} (1 - c_{i,j-1}), \quad (251)$$

where j is the number of a neuron in a following layer and r represents an excitatory rate in a neuron [370]. Reid and Ferens used $r = 4$ as this value produces a chaotic behavior of the logistic map [370]; generally, values below 0 or above 4 lead to the output to become unbounded, values between 0 and 1 lead to convergence toward the zero, values between 1 and 3 lead to convergence to a fixed number, values between 3 and 3.5 lead to a periodic solution and only values between 3.5 and 4 produce chaotic behavior [370].

3.48.2 Fusion of chaotic activation function (FCAF)

Similarly as HCAF, also the fusion of chaotic activation function (FCAF) [371] uses a recursive definition for computing the output of a neuron. The FCAF is defined³⁸ for hidden units as³⁹

$$f(z_{i+1}) = r z_i (1 - z_i) + z_i + a - \frac{b}{2\pi} \sin(2\pi z_i) \quad (252)$$

and for the output units as

$$f(z_{i+1}) = r z_i (1 - z_i) + z_i + a - \frac{b}{2\pi} \sin(2\pi z_i) + \exp(-c z_i^2) + d, \quad (253)$$

where r , a , b , c , and d are fixed parameters [371]; the suitable values for the parameter r are discussed in section 3.48.1 where an equivalent parameter is used.

3.48.3 Cascade chaotic activation function (CCAF)

The cascade chaotic activation function (CCAF) was introduced in [372] and is recursively defined for the neuron $i + 1$ in a given layer using the preceding neuron i from the same layer as

$$f(z_{i+1}) = a \cdot \sin(\pi \cdot b \cdot \sin(\pi z_i)), \quad (254)$$

where a and b are two fixed parameter from the interval $[0, 1]$ [372].

4 Adaptive activation functions

The activation function introduces non-linearities to neural networks and is crucial for network's performance [136]. Even though it might be suboptimal, the same activation function is usually used for the whole network or at least for all neurons in a single layer. Over the last few decades, there have been several attempts to use activation functions that might differ across neurons (e.g., [233, 373–376]). The adaptive activation functions — i.e., functions that have a trainable parameter that changes their shape — have been receiving more attention recently (e.g., [233, 377–379]) and might become a new standard in the field. One of the first descriptions of the general AAF approach is available in [373] where Wu, Zhao, and Ding described an AF⁴⁰ that has one or more trainable parameters that are trained together with the rest of the network's weights [373]. The simplest forms just add a parameter to a particular neural network that controls one of its properties (e.g., slope), while the more complex ones allow for the learning of a large number of activation functions (e.g., adaptive spline activation functions in [376]).

³⁸Kabir et al. did not explicitly defined what the index i denotes but most likely it denotes the i -th neuron in a given layer.

³⁹The formula given in [371] probably missed a minus sign after the parameter a .

⁴⁰The authors used the name trainable activation function (TAF) rather than the adaptive activation function (AAF) that is used throughout this work.

4.1 Transformative adaptive activation function (TAAF)

The transformative adaptive activation function (TAAF) [380, 381] is a family of AFs that adds four adaptive parameters for scaling and translation of any given AF — as such, the TAAFs represent a simple framework with a small set of additional parameters that generalizes a lot of AAFs that are listed in this work. While there are even more general approaches such as the adaptive blending unit (ABU) (see section 4.48) that allows for a combination of several different activation functions, the TAAFs are conceptually simpler and add only four additional parameters. The TAAF is defined as

$$g(f, z_i) = \alpha_i \cdot f(\beta_i \cdot z_i + \gamma_i) + \delta_i, \quad (255)$$

where z_i input to the AF, α_i , β_i , γ_i , and δ_i , are adaptive parameters for each neuron i [380]. Therefore, the output of a neuron with TAAF with inputs x_i is:

$$\alpha_i \cdot f\left(\beta_i \cdot \sum_{i=1}^n w_i x_i + \gamma_i\right) + \delta_i, \quad (256)$$

where x_i are individual inputs, w_i are its weights, and n is the number of incoming connections [380]. If there is no unit x_i , then the parameter γ is equivalent to the bias term of the neuron [380]. It was shown in [380] that each of the four adaptive parameters statistically significantly improve the performance of the AF — this is not surprising as many of the AFs presented in this work use subset of these parameters. For example, equivalent of α is used in the positive PReLU (see section 4.2.2), equivalent of β in the swish (see section 4.4.1), and equivalent of γ and δ in the FReLU (see section 4.2.15).

4.2 The ReLU-based family of adaptive functions

There are numerous ReLU extensions that are adaptive [1]. Some of the adaptive activations have a non-adaptive counterpart — e.g., PReLU (see section 4.2.1), which is basically a LReLU with an adaptive parameter of leakiness.

4.2.1 Parametric rectified linear unit (PReLU)

However, AAFs might be very useful even in the simplest form with a single added parameter — an AAF called PReLU was used to obtain a state-of-the-art result on the ImageNet Classification in 2015, the first surpassing human-level performance [233]. The PReLU generalizes the ReLU by adding a parameter that controls the slope of the activation function for negative inputs (the ReLU is constant at zero for negative inputs) that is learned with other weights [382]:

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ \frac{z_i}{a_i}, & z_i < 0, \end{cases} \quad (257)$$

where a_i is an optimized parameter for each neuron/filter i . The LReLU [231] is essentially a PReLU but with the parameter a_i fixed and not trainable (see section 3.6.2 for LReLU details). PReLUs are better than ReLUs for verification-friendly NNs [383].

4.2.2 Positive parametric rectified linear unit (PReLU⁺)

The positive PReLU is an adaptive variant of the SiReLU (see section 3.6.5) proposed in [384]; it is also a special case of, for example, DPRELU, Dual Line, and piecewise linear unit (PiLU). It is defined as

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ 0, & z_i < 0, \end{cases} \quad (258)$$

where a_i is a trainable parameter [384].

4.2.3 Margin Relu

The margin (MarReLU)⁴¹ is an adaptive variant of the Shifted ReLU where the shift a_i is determined as the channel-by-channel expectation value of the negative response [385]. It is defined as

$$f(z_i) = \max(z, a_i) = \begin{cases} z, & z - a_i \geq 0, \\ a_i, & z - a_i < 0. \end{cases} \quad (259)$$

⁴¹Heo et al. abbreviated it as MReLU, but this abbreviation is used for the mirrored rectified linear unit (see section 3.6.28) in this work.

4.2.4 Funnel parametric rectified linear unit (FunPReLU)

The funnel rectified linear unit (FunReLU)⁴² and funnel (FunPReLU) are 2D AFs proposed in [386]. The FunReLU and FunPReLU introduce a spatial context into the AF by comparing the input to a funnel condition instead of the zero that is used as the threshold in ReLU and PReLU [386]. The FunReLU is defined as

$$f(z_{c,m,n}) = \max(z_{c,m,n}, t(z_{c,m,n})), \quad (260)$$

where $z_{c,m,n}$ is the input on the c -th channel at the 2D spatial position m, n and $t(z_{c,m,n})$ is the spatial context from a 3×3 window⁴³

$$t(z_{c,m,n}) = \sum_{m-1 \leq h \leq m+1, n-1 \leq w \leq n+1} z_{c,h,w} \cdot p_{c,h,w} \quad (261)$$

and $p_{c,h,w}$ denotes the coefficients on this window [386]. The FunPReLU is defined similarly. The FunReLU was, for example, used in [387, 388].

4.2.5 React-PReLU (RReLU)

The react- (RReLU) is an adaptive variant of the PReLU with vertical and horizontal shifts; it is defined as

$$f(z_i) = \begin{cases} z_i - a_c + b_c, & z_i \geq a_c, \\ c_c(z_i - a_c) + b_c, & z_i < a_c, \end{cases} \quad (262)$$

where a_c , b_c , and c_c are trainable parameters for each channel c and z_i denotes the input to the neuron i in the channel c [389]; a_c controls the horizontal shift, b_c controls the vertical shift, and c_c is the slope parameter for negative inputs as in the original PReLU.

4.2.6 Smooth activation unit (SAU)

The smooth activation unit (SAU) is a smoothed variant of the PReLU⁴⁴ using the convolution operation with the Gaussian function [390]. It is defined as

$$f(z_i) = (\text{PReLU}_{a_i} * \phi_{b_i})(z_i), \quad (263)$$

where $*$ is the convolution operation, PReLU_{a_i} is the PReLU⁴⁵ parameterized by a_i ⁴⁶ and $\phi_{b_i}(x)$ is the Gaussian function parameterized by b_i inversely controlling the deviation of the function [390]. The resulting AF is then

$$f(z_i) = \frac{1}{2b_i} \sqrt{\frac{2}{\pi}} \exp\left(-\frac{b_i^2 z_i^2}{2}\right) + \frac{1 + \frac{1}{a_i}}{2} z_i + \frac{1 - \frac{1}{a_i}}{2} z_i \cdot \text{erf}\left(\frac{b_i z_i}{\sqrt{2}}\right), \quad (264)$$

where a_i and b_i are either fixed or trainable parameters [390].

4.2.7 Smooth maximum unit (SMU)

The smooth maximum unit (SMU) [391] is an AAF that uses a smooth approximation of the absolute value function. The SMU is defined as

$$f(z_i) = \frac{(1 + a_i) z_i + (1 - a_i) z_i \text{erf}(b_i (1 - a_i) z_i)}{2} \quad (265)$$

where a_i and b_i are learnable parameters [391]. This smooth approximation of the absolute value function using the Gaussian error function could be used to create a whole class of AFs similarly as in section 4.55.

4.2.8 Leaky Learnable ReLU (LeLeLU)

An adaptive LReLU variant named leaky learnable ReLU (LeLeLU) was proposed in [392]. It is a LReLU with learnable scaling parameter:

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ 0.01 a_i z_i, & z_i < 0, \end{cases} \quad (266)$$

where a_i is a trainable parameter for each neuron i [392].

⁴²Ma, Zhang, and Sun originally named the unit FReLU but its abbreviation would collide with the flexible ReLU.

⁴³Other sizes were also tested in [386] but Ma, Zhang, and Sun found 3×3 to work the best.

⁴⁴The principle could be, however, applied to other AFs.

⁴⁵Biswas et al. used the LReLU in the definition of the SAU but since they consider the parameter a_i trainable, we stick to the usage of PReLU in the definition.

⁴⁶To conform to the used definition of the PReLU unit, we will use the slope scaling by $\frac{1}{a_i}$ even though authors originally used the a_i for slope scaling of negative inputs.

4.2.9 Parametric rectified exponential unit (PREU)

Similarly as PReLU extends the ReLU (see section 4.2.1), the PREU extends the swish and ELU inspired REU [296]. It is defined as

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ a_i z_i \cdot \exp(b_i z_i), & z_i < 0, \end{cases} \quad (267)$$

where a_i and b_i are trainable parameters for each neuron/filter i [296]. The advantage of PREU is that it uses the negative information near zero [1] — unlike the ReLU.

4.2.10 Randomly translational PReLU (RT-PReLU)

A randomly translational PReLU (RT-PReLU) an equivalent extension to PReLU as is RT-ReLU to ReLU (see section 3.6.11) [246]. It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i + b_i \geq 0, \\ \frac{z_i}{a_i}, & z_i + b_i < 0, \end{cases} \quad (268)$$

where a_i is a trainable parameter and b_i is a stochastic parameter for each neuron i randomly sampled from the Gaussian distribution at each iteration, $b_i \sim N(0, \sigma^2)$, where σ^2 is the variance of the Gaussian distribution. The offset b_i is set to zero during the test phase [1]. The authors Cao et al. set the $\sigma^2 = 0.75^2$ for their experiments [246]. It is also possible to have the parameter b_i sampled for each neuron i , but the a_i is shared by all neurons in a channel c [393].

4.2.11 Probabilistic activation (ProbAct)

A probabilistic class of activation functions ProbAct that adds a random noise to any activation function [394]. It is defined as

$$f(z) = g(z) + \sigma e, \quad (269)$$

where $g(z)$ is any function (either fixed or trainable) defining the mean of the probabilistic activation, $e \sim N(0, 1)$ is a random value sampled from a standard normal distribution, and σ is either fixed or learnable parameter controlling the range of the perturbation [394]. σ can be either a global learnable parameter or different for each neuron i [394]. The ProbAct used in [394] is a ReLU based ProbAct defined as

$$f(z) = \max(0, z) + \sigma e, \quad (270)$$

which resembles NReLU (see section 3.6.6) that adds random noise for output values for the positive inputs z [221]. A similar concept for sigmoid and tanh activation was used in [395] (see section 4.18).

The chaotic injections presented in [396] represent a similar approach; however, the injections are not stochastic but rather defined using the chaos theory. Furthermore, Reid, Ferens, and Kinsner discuss several approaches for injections of a chaotic value s_n into a ReLU: $\text{ReLU}(z + s_n)$, $\text{ReLU}(z \cdot s_n)$, $\text{ReLU}(z + z \cdot s_n)$, $\text{ReLU}(z) + s_n$, $\text{ReLU}(z) \cdot s_n$, $\text{ReLU}(z) + z s_n$, and $\text{ReLU}(z) + \text{ReLU}(z) \cdot s_n$ [396].

4.2.12 Adaptive offset activation function (AOAF)

Another ReLU variant with adaptive shift termed adaptive offset activation function (AOAF) was defined in [397]. The AOAF introduces two hyperparameters and one data-dependent adaptive parameter; it is defined as

$$f(z_i) = \max(0, z_i - b a_i) + c a_i, \quad (271)$$

where b and c are predefined, fixed parameters and a_i is the mean value of the inputs of neuron i [397]. The recommended values for the parameters b and c are $b = c = 0.17$ as it yielded the best image classification accuracy in the experiments [397].

4.2.13 Dynamic leaky ReLU (DLReLU)

An error based dynamic leaky ReLU (DLReLU) was proposed in [398] (under the name of Dynamic ReLU — DReLU — but this naming collides with DReLU established in [399, 400]; see section 4.2.14 and section 4.56.3). The DLReLU is a LReLU where the leakiness depends on the test error from the previous epoch [1]

$$f(z) = \begin{cases} z, & z \geq 0, \\ ab_t z, & z < 0, \end{cases} \quad (272)$$

where $a \in (0, 1)$ is a predefined parameter controlling the leakiness similarly as in LReLU (see section 3.6.2) and b_t is a dynamic parameter computed for current training epoch t as the test error from the previous epoch $t - 1$, i.e., $b_t = \text{MSE}_{t-1}$ [398].

A version exp-DLReLU was proposed to deal with deeper networks with more than seven hidden layers in order to avoid too large error values causing the training to fail [398]:

$$f(z) = \begin{cases} z, & z \geq 0, \\ ac_t z, & z < 0, \end{cases} \quad (273)$$

where $c_t = \exp(-b_t) = \exp(\text{MSE}_{t-1})$ [398].

The advantage of DLReLU and exp-DLReLU is that the changes in leakiness are big at the beginning of the training due to higher test error and diminish towards the end [398]. A similar effect could be obtained by a schedule of the leakiness parameter in the LReLU.

4.2.14 Dynamic ReLU (DReLU)

Similar approach to the ABReLU is presented by the DReLU [399] (not to be confused with identically named activations in [398, 400]),

$$f(z_i) = \begin{cases} z_i, & z_i - a_i \geq 0, \\ a_i, & z_i - a_i < 0, \end{cases} \quad (274)$$

where a_i is a threshold value that is computed as the midpoint of the range of input values for each batch; e.g., if the values range from -4 to 8, then $a_i = \frac{-4+8}{2} = 2$ [399]. The DReLU can be considered to be a variant of DisReLU (see section 3.6.44) with data-dependent determination of the shifting point.

4.2.15 Flexible ReLU (FReLU)

The FReLU is a ReLU extension with zero-like property and the ability to capture negative information [401]. The zero-like property is the ability to push activation means closer to zero [401] as this might speed up learning [236]. The FReLU builds on the ability to shift the AF

$$f(z_i) = \text{ReLU}(z_i + a_i) + b_i, \quad (275)$$

where a_i and b_i would be optimized parameters [401]. However, since the parameter a_i can be learned by the bias term of the neuron to whose output is the activation function applied, the authors Qiu, Xu, and Cai formulate the FReLU as

$$f(z_i) = \text{ReLU}(z_i) + b_i = \begin{cases} z_i + b_i, & z_i \geq 0, \\ b_i, & z_i < 0, \end{cases} \quad (276)$$

where b_i is a trainable parameter [401].

4.2.16 Adaptive shifted ReLU (ShiLU)

An adaptive shifted ReLU (ShiLU) [252] is another adaptive variant of the ReLU activation; it is a variant that adds a trainable slope and a vertical shift:

$$f(z_i) = a_i \text{ReLU}(z_i) + b_i = a_i \cdot \max(0, z) + b_i, \quad (277)$$

where a_i and b_i are trainable parameters for each neuron i [252]. They used the name *shifted ReLU*, but that name is already taken by the non-adaptive Shifted ReLU; hence, the full name is adaptive shifted ReLU throughout this work to avoid confusion.

4.2.17 StarReLU

The StarReLU [402] is an adaptive version of the RePU of power 2 using a similar approach as the ShiLU; it is defined as

$$f(z_i) = a_i (\text{ReLU}(z))^2 + b_i, \quad (278)$$

where a_i and b_i are trainable parameters for each neuron i [402]. If the parameters are not used in an adaptive manner, Yu et al. recommend setting $a_i = 0.8944$ and $b_i = -0.4472$ [402].

4.2.18 Adaptive HardTanh

An adaptive variant of HardTanh was used in [257]; it is defined as

$$f(z) = \text{HardTanh}(a_t(z - b)), \quad (279)$$

where a_t is a scale factor for each epoch t such that $1 \leq a_1 \leq a_2 \leq \dots \leq a_t \leq \dots \leq a_T$, T is the total number of training epochs and b is an adaptive parameter trained using BP with other parameters of the NN [257]. The parameters a_t are set such that the function starts in a similar shape as a regular HardTanh (see section 3.6.18) and gradually approaches the sign function [257]. This allows for training a network that will gradually become a binary NN where each activation is the sign function which can be used for speeding the inference [257].

4.2.19 Attention-based ReLU (AReLU)

Attention-based ReLU (AReLU) [403] is a adaptive ReLU variant that uses ELSA — element-wise attention mechanism proposed in [403]. It is defined as

$$f(z_l) = \begin{cases} (1 + \sigma(b_l)) z_l, & z_l \geq 0, \\ C(a_l) z_l, & z_l < 0, \end{cases} \quad (280)$$

where a_l and b_l are learnable parameters for each layer l , $\sigma(x)$ is the logistic sigmoid function, $C(x)$ is a function that clips the input into $[0.01, 0.99]$ [403]. The derivative of $C(a_l)$ is handled by just not using the BP when $a_l < 0.01$ or $a_l > 0.99$ [403]. While Chen, Li, and Xu observe that the parameters a_l and b_l are insensitive to the initialization, they recommend initializing $a_l = 0.9$ and $b_l = 2.0$ as a larger initial value of b_l can speed up the convergence [403]. The AReLU was found to outperform CELU, ELU, GELU, LReLU, Maxout, Relu, RReLU, SELU, sigmoid, softplus, swish, tanh, adaptive piece-wise linear unit (APLU), Padé activation unit (PAU), PReLU, and self-learnable activation function (SLAF) in experiments with various learning rates in [403]. The performance of AReLU was validated under different settings in [63, 177, 404].

4.2.20 Dual parametric ReLU (DPRReLU) and Dual Line activation function

A DPRReLU [405] extends the concept of PReLU even further:

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ b_i z_i, & z_i < 0, \end{cases} \quad (281)$$

where a_i and b_i are trainable parameters for each neuron i ; these are initialized the same as PReLU — $a_i = 1$, $b_i = 0.01$ [405]. The DPRReLU was also later proposed independently in [406] under the name *fully parametric ReLU* and the abbreviation FReLU (which is already used in the literature for the flexible ReLU; see section 4.2.15).

4.2.21 Dual Line

The DPRReLU was further extended into a Dual Line activation function that adds a shift parameter

$$f(z_i) = \begin{cases} a_i z_i + m_i, & z_i \geq 0, \\ b_i z_i + m_i, & z_i < 0, \end{cases} \quad (282)$$

where a_i and b_i are trainable parameters for each neuron i the same as in DPRReLU and m_i is an additional trainable shift parameter for each neuron or filter i ; m_i was initialized to $m_i = -0.22$ [405].

4.2.22 Piecewise linear unit (PiLU)

An AF very similar to the Dual Line is the piecewise linear unit (PiLU) proposed in [407]; it just extends the Dual Line concept by adding horizontal shifts. It is defined as

$$f(z_i) = \begin{cases} a_i z_i + c_i(1 - a_i), & z_i \geq c_i, \\ b_i z_i + c_i(1 - b_i), & z_i < c_i, \end{cases} \quad (283)$$

where a_i , b_i , and c_i are adaptive parameters for each neuron i [407]. The PiLU generalizes, for example, ReLU, LReLU, PReLU, SiReLU, DPRReLU, and Dual Line.

4.2.23 Dual parametric family of activation functions

The DPRELU approach (see section 4.2.20) can be extended to a general concept transforming any activation function $g(z)$:

$$f(z_i) = \begin{cases} a_i g(z_i) + m_i, & z_i \geq 0, \\ g(z_i) + m_i, & z_i < 0, \end{cases} \quad (284)$$

where $g(z_i)$ is any activation function and a_i and m_i are trainable parameters for each neuron i [405]. The functions of this family are called dual parametric activation functions (DPAFs) throughout this text.

4.2.24 Fully parameterized activation function (FPAF)

Similar approach to DPAF (see section 4.2.23) was proposed under the name of fully parameterized activation function (FPAF) in [406]; the FPAF is defined as

$$f(z_i) = \begin{cases} a_i g_1(z_i), & z_i \geq 0, \\ b_i g_2(z_i), & z_i < 0, \end{cases} \quad (285)$$

where a_i and b_i are trainable parameters for each neuron i and $g_1(z_i)$ and $g_2(z_i)$ can be any function [406]. The FPAF, in contrast to the family of DPAFs, has no trainable shift but allows for learnable slopes for both parts of the piecewise function.

4.2.25 Elastic PReLU (EPReLU)

The same as EReLU extends the concept of ReLU (see section 3.6) [281], the Elastic (EPReLU) extends the PReLU — it adds a varying coefficient to the positive part of the PReLU [281]:

$$f(z_i) = \begin{cases} k_i z_i, & z_i \geq 0, \\ \frac{z_i}{a_i}, & z_i < 0, \end{cases} \quad (286)$$

where a_i is the optimized parameter, k_i is a sampled for each epoch and neuron i from the uniform distribution: $a_i \sim U(1 - \alpha, 1 + \alpha)$ where $\alpha \in (0, 1)$ [281]. A modified training procedure for EPReLU is also proposed — the neuron weights and the trainable parameter a_i are updated with $k_i = 1$ in odd epochs, while in even epochs, the a_i is kept fixed, the parameter k_i is sampled from the uniform distribution, and only the neuron weights are updated [281]. It was shown that the EPReLU leads to improved performance over the ReLU, PReLU, EReLU, APLU, network in network (NIN), and maxout unit networks on several datasets [281].

4.2.26 Paired ReLU

A paired ReLU [408] is a concept similar to CReLU (see section 3.6.34), but it introduces four trainable parameters. It is defined as

$$f(z) = \begin{bmatrix} \max(a_i z_i - b_i, 0) \\ \max(c_i z_i - d_i, 0) \end{bmatrix}, \quad (287)$$

where a_i, b_i, c_i , and d_i are trainable parameters for each neuron i [1, 408]. The parameters a_i and c_i are scale parameters and b_i and d_i are trainable thresholds; the initial values of scale parameters are $a_i = 0.5$ and $c_i = -0.5$ [408].

4.2.27 Tent

The tent is a ReLU-based AF proposed in [409]; it is defined as

$$f(z_i) = \max(0, a_i - |z_i|), \quad (288)$$

where a_i is a trainable parameter [409]. Rozsa and Boulton recommend using batch normalization and initializing $a_i = 1$ [409]. Also, having a weight decay on the parameter a_i during training proved beneficial for certain tasks [409].

4.2.28 Hat

The hat [410] is an AAF very similar to the tent AF — the only difference is that the tent AF is centered around zero while the hat is positive only for positive inputs. The hat AF is defined as

$$f(z_i) = \begin{cases} 0, & z_i < 0, \\ z_i, & 0 \leq z_i \leq \frac{a_i}{2}, \\ z_i, & \frac{a_i}{2} \leq z_i \leq a_i, \\ 0, & z_i > a_i, \end{cases} \quad (289)$$

where a_i is can be either fixed or trainable parameter [410]. Wang, Xu, and Zhu used $a_i = 2$ for the fixed variant in [410]; this value was also used in [411].

4.2.29 ReLU memristor-like activation function (RMAF)

The ReLU memristor-like activation function (RMAF) is an activation function similar to the swish AF (see section 4.4.1) and it also attempts to leverage the negative values [412]. It is defined as

$$f(z_i) = \left[b \frac{1}{(0.25(1 + \exp(-z_i)) + 0.75)^c} \right] a_i z_i, \quad (290)$$

where a_i is a trainable parameter initialized $a_i = 1$ for each neuron i or it is a fixed hyperparameter and b and c are fixed hyperparameters [412].

4.2.30 Parametric tanh linear unit (PTELU)

A parametric tanh linear unit (PTELU) [413] is an adaptive function that, for positive inputs, behaves just as a ReLU; however, the negative part is parameterized tanh function [1]. It can also be seen as an extension of the PReLU (see section 4.2.1). It is an adaptive variant of ThLU (see section 3.7.1). It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i \tanh(b_i z_i), & z_i < 0, \end{cases} \quad (291)$$

where a_i and b_i are trainable parameters for each neuron i ; $a_i \geq 0$ and $b_i \geq 0$ [413]. It has output range of $[-a_i, \infty)$ [1]. The parameter a_i controls the saturation value, and the parameter b_i controls the convergence rate [413]. While the AF resembles an adaptive extension of an ELU activation functions, the author Gupta and Duggal decided to use tanh function for the negative inputs because it gives a higher gradient for small negative inputs and saturates earlier than $\exp(z) - 1$ and thus the noise-robust deactivation state earlier and faster [413]. The nonadaptive variant of PTELU with $a_i = 1$ and $b_i = 1$ was proposed in [170].

4.2.31 Tangent linear unit (TaLU)

The tanh linear unit (TaLU) [414] is an AF similar to the PTELU. The TaLU is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ \tanh(z_i) & a_i < z_i < 0, \\ \tanh(a_i) & z_i \leq a_i, \end{cases} \quad (292)$$

where $a_i < 0$ is either a learnable⁴⁷ or fixed parameter [414].

4.2.32 PTaLU

The PTaLU⁴⁸ is a variant of TaLU with another learnable parameter [414]. It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq b_i, \\ \tanh(z_i) & a_i < z_i < b_i, \\ \tanh(a_i) & z_i \leq a_i, \end{cases} \quad (293)$$

where a_i and b_i are trainable parameters [414]. Mercioni and Holban used initial values $a_i = -0.75$ and $b_i = 1$ in [414].

4.2.33 TanhLU

The tanhLU⁴⁹ is a parametric combination of the tanh and a linear function proposed in [415]. It is defined as

$$f(z_i) = a_i \cdot \tanh(b_i z_i) + c_i z_i, \quad (294)$$

where a_i , b_i , and c_i are trainable parameters for each neuron i [415].

⁴⁷The variant with the adaptive parameter was named *TaLU learnable* by the authors.

⁴⁸Not an abbreviation but a name given by Mercioni and Holban in [414].

⁴⁹Not an abbreviation but a name given by Shen et al. in [415].

4.2.34 TeLU

Despite the similar name, the tanh exponential linear unit (TeLU)⁵⁰ [204] is quite different from the PTELU. The TeLU is closely related to the mish and TanhExp activations, but, unlike these two AFs, it also has an additional adaptive parameter. It is defined as

$$f(z_i) = z_i \cdot \tanh(\text{ELU}(a_i z_i)), \quad (295)$$

where a_i is either learnable or fixed scaling parameter [204].

4.2.35 Tanh based ReLU (TReLU)

A TReLU was proposed in [416]; however, it is only a special case of previously proposed PTELU (see section 4.2.30). It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ \tanh(b_i z_i), & z_i < 0, \end{cases} \quad (296)$$

where b_i is a trainable parameter for each neuron i [416]. This function is identical to the PTELU with its parameter a_i fixed to $a_i = 1$. Another special case of PTELU was proposed in [417] also under the name of TReLU — this time, the parameter a_i becomes a predefined fixed parameter, and b_i becomes fixed to $b_i = 1$. This function is denoted TReLU variant 2 (TReLU2) in this work and is defined as

$$f(z) = \begin{cases} z, & z \geq 0, \\ a \tanh(z), & z < 0, \end{cases} \quad (297)$$

where a is fixed⁵¹ parameter [417].

4.2.36 Rectified linear tanh (ReLTanh)

A ReLTanh is a piecewise adaptive activation function that improves traditional tanh activation function [418] — it replaces the positive and negative saturated regions of the tanh activation functions with straight lines whose slopes are identical to the slope of the tanh at the thresholds [418]. It is defined as

$$f(z_i) = \begin{cases} \tanh'(a_i)(z_i - a_i) + \tanh(a_i), & z_i \leq a_i, \\ \tanh(z_i), & a_i < z_i < b_i, \\ \tanh'(b_i)(z_i - b_i) + \tanh(b_i), & z_i \geq b_i, \end{cases} \quad (298)$$

where $\tanh'(x)$ is the derivative of the tanh function

$$\tanh'(x) = \frac{4}{(\exp(x) + \exp(-x))^2}, \quad (299)$$

and $a_i \in [a_{\text{low}}, a_{\text{high}}]$ and $b_i \in [b_{\text{low}}, b_{\text{high}}]$ are two trainable parameters that may be defined for each neuron i but are rather recommended to be shared by a whole layer l [418] in order to decrease computational burden. The limits a_{low} , a_{high} , b_{low} , and b_{high} for the parameters are to constraint the learnable range and are predefined hyperparameters. Wang et al. used $a_{\text{low}} = -\infty$, $a_{\text{high}} = -1.5$, $b_{\text{low}} = 0$, and $b_{\text{high}} = 0.5$ in their work [418]. The initial values were set to $a_i = -1.5$ and $b_i = 0$ for all layers (the parameters were shared layer-wise) in order to speed up the training process in early stages by the larger gradients [418].

4.2.37 Bendable linear unit (BLU)

A BLU [419] is an adaptive function that allows for any interpolation between the identity function and a rectifier [1, 419]. It is defined as

$$f(z_i) = a_i \left(\sqrt{z_i^2 + 1} - 1 \right) + z_i, \quad (300)$$

where $a_i \in [-1, 1]$ is a trainable parameter for each neuron or filter [419]. One of the main advantages of the BLU is that it can model an identity function; the identity function is useful because its gradient cannot vanish or explode, and it also allows for a layer to be "skipped" [419] — it is one of the reasons of why ResNets [196] became so popular

⁵⁰[204] used the TeLU as the name and not as an abbreviation; nevertheless, the long name tanh exponential linear unit fits the usual naming convention and, therefore, it is used in this work.

⁵¹While Nakhua et al. used the parameter fixed during their experiments, they also speculated that making it learnable might improve the performance.

[419] as it is rather hard to learn an identity transformation using conventional neural network and the architecture with skip connections allows for easy learning of the identity mapping [196]. Unless the magnitude $|a_i|$ is exactly 1, the derivative of BLU is non-zero for both positive and negative inputs (similarly to LReLU and in contrast to vanilla ReLU and ELU) [419]. BLU has a slope higher than 1 for positive inputs for a_i approaching 1 (or for negative inputs for a_i approaching -1) [419]; this property helps to avoid vanishing gradient problems [309, 419]. Another useful benefit is that BLU are C^∞ continuous [419], which can be theoretically exploited for speeding up the optimization [419], e.g., [420–423]. It was also shown that smooth activation functions provide better signal propagation [424].

4.2.38 Rectified BLU (ReBLU)

A variant of the BLU (see section 4.2.37) was proposed in [384] under the name rectified BLU (ReBLU). It is defined as

$$f(z_i) = \begin{cases} \text{BLU}(z_i), & z_i > 0, \\ 0, & z_i \leq 0, \end{cases} = \begin{cases} a_i \left(\sqrt{z_i^2 + 1} - 1 \right) + z_i, & z_i > 0, \\ 0, & z_i \leq 0, \end{cases} \quad (301)$$

, where a_i is a trainable parameter [384].

4.2.39 DELU

The DELU⁵² activation function [252] is a ReLU variation that utilizes the SiLU function (see section 3.3). It is defined as

$$f(z_i) = \begin{cases} (a_i + 0.5)z_i + |\exp(-z_i) - 1|, & z_i \geq 0, \\ z_i \sigma(z_i), & z_i < 0, \end{cases} \quad (302)$$

where a_i is a trainable parameter for each neuron i and $\sigma(z_i)$ is the logistic sigmoid function [252].

4.2.40 Soft clipping mish

A ReLU variant called soft clipping mish (SC-mish) was proposed in [425]. It adds soft clipping to the positive inputs using the mish AF; it is defined as

$$f(z_i) = \max(0, z_i \cdot \tanh(\text{softplus}(a_i z_i))), \quad (303)$$

where a_i is a fixed parameter [425]; Mercioni and Holban used $a_i = 1$. It also has a variant where the parameter a_i is trainable. Such a variant is called soft clipping learnable mish (SCL-mish). When using the SCL-mish, Mercioni and Holban initialized the parameter $a_i = 0.25$ [425].

4.2.41 Soft clipping swish

Yet another AF proposed by Mercioni and Holban is the soft clipping swish (SC-swish) [426–428]. This function is very similar to SC-mish and is defined as

$$f(z) = \max(0, z \cdot \sigma(z)), \quad (304)$$

where $\sigma(z)$ is the logistic sigmoid [426].

4.2.42 Parametric swish (p-swish)

The parametric swish (p-swish) [429] is another AF proposed by Mercioni and Holban. It is defined as

$$f(z_i) = \begin{cases} a_i z_i \sigma(b_i z_i), & z_i \leq c_i, \\ z_i, & z_i > c_i, \end{cases} \quad (305)$$

where a_i , b_i , and c_i are either trainable or fixed parameters (or combination thereof) [429]. The parameters were initialized to $a_i = 1$, $b_i = 1$ and $c_i = 0$ in experiments in [429]. An AF named R_S similar to the p-swish was independently proposed in [430]; it is equivalent to a p-swish with fixed $a_i = 1$.

⁵²DELU is not an abbreviation but rather a name given by Pishchik.

4.2.43 Parametric exponential linear unit (PELU)

Similarly as PReLU extends the concept of ReLU, the parametric exponential linear unit (PELU) [431] extends the concept of ELU (see section 3.6.48). The PELU builds on a parameterization that separately controls the saturation point, the decay, and the slope:

$$f(z_i) = \begin{cases} c_i z_i, & z_i \geq 0, \\ a_i \left(\exp\left(\frac{z_i}{b_i}\right) - 1 \right), & z_i < 0, \end{cases} \quad (306)$$

where a_i , b_i , and c_i are trainable parameter for each neuron i [431]. However, the PELU introduces only two new parameters — a_i controlling the saturation point, and b_i controlling the decay — to control the shape of the activation function; the slope is not controlled separately through another parameter as it could lead to non-differentiability at $z_i = 0$; therefore the slope is set such that the derivatives on both sides of zero are equal which leads to $c_i = \frac{a_i}{b_i}$ [431] and therefore the PELU is defined as

$$f(z_i) = \begin{cases} \frac{a_i}{b_i} z_i, & z_i \geq 0, \\ a_i \left(\exp\left(\frac{z_i}{b_i}\right) - 1 \right), & z_i < 0. \end{cases} \quad (307)$$

The PELU combined with mixing different activation functions which use an adaptive linear combination or hierarchical gated combination of activation function was shown to perform well [432] — see section 4.2.45.

4.2.44 Extended exponential linear unit (EDELU)

An adaptive function called extended exponential linear unit (EDELU)⁵³ was proposed in [433]. This function is the same as the PELU, but it omits the vertical scaling for positive inputs, adds a parameter controlling the threshold, and uses inverse definitions of the parameters present in the PELU. It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq c_i, \\ \frac{\exp(a_i z_i) - 1}{b_i}, & z_i < c_i, \end{cases} \quad (308)$$

where $a_i \geq 0$ ⁵⁴ and $b_i \geq 0$ ⁵⁵ are trainable PELU parameters and $c_i \geq 0$ is the novel parameter for controlling the threshold that has to satisfy the relationship

$$b_i c_i = \exp(a_i c_i) - 1; \quad (309)$$

while the $c_i = 0$ is always a solution of the equation, there are other solutions for $b_i > a_i > 0$ [433].

4.2.45 Adaptive combination of PELU and PReLU

Two different activation functions can be mixed together, as shown in [432]. One such example of mixed activation function is

$$f(z_i) = a_i \cdot \text{LReLU}(z_i) + (1 - a_i) \text{ELU}(z_i), \quad (310)$$

where a_i is a combination coefficient that might be learned from the data [432]. Another mixing approach was shown for combining PReLU and PELU:

$$f(z_i) = \sigma(a_i z_i) \text{PReLU}(z_i) + (1 - \sigma(a_i z_i)) \text{PELU}(z_i), \quad (311)$$

where $\sigma(x)$ is the logistic sigmoid [432]. Qian et al. also proposed other mixing schemes such as hierarchical activation, winner-take-all selection whose performance were shown on the MNIST [182], CIFAR-10 and CIFAR-100 [158] datasets; see [432] for details.

4.2.46 Fast exponential linear unit (FELU)

An ELU variant called fast exponential linear unit (FELU) aiming at efficient training and network inference was proposed in [434] — it is inspired by fast approximation of the exponential function proposed in [435] to replace the exponential in the ELU:

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i \left(2^{\frac{z_i}{\ln(2)}} - 1 \right), & z_i < 0, \end{cases} \quad (312)$$

where a_i is a trainable parameter controlling the soft saturation region [434].

⁵³Authors called the function *extended ELU* resulting in an abbreviation DELU but that name is already taken by an AF proposed a few months earlier in [252].

⁵⁴The PELU equivalent would be $\frac{1}{b_i}$.

⁵⁵The PELU equivalent would be $\frac{1}{a_i}$.

4.2.47 P+FELU

Adem proposed variant of the FELU function named P+FELU; this variant has an added parameter and is defined as

$$f(z_i) = \begin{cases} z_i + b, & z_i \geq 0, \\ a_i \left(2^{\frac{z_i}{\ln(2)}} - 1 \right) + b, & z_i < 0, \end{cases} \quad (313)$$

where a_i is a trainable parameter same as the original FELU and b is the added trainable parameter [436].

4.2.48 Multiple parametric exponential linear unit (MPELU)

A PELU extension, multiple parametric exponential linear unit (MPELU) [437], uses two trainable parameters to allow for a combination of a ReLU and ELU [1]. The multiple parametric exponential linear unit (MPELU) is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i (\exp(b_i z_i) - 1), & z_i < 0, \end{cases} \quad (314)$$

where a_i and b_i are trainable parameters for each neuron i [1, 437]. The ReLU, certain parameterizations of PELU, and ELU are special cases of the MPELU [437]. A special method for weight initialization of neurons with MPELU units was also proposed; depending on a particular initialization, the method can become the initialization for ELU networks or for ReLU networks [437]. The MSRA⁵⁶ filler approach [233] can be considered as a special case of the MPELU initialization [437]. The MPLU initialization is a similar approach to LSUV initialization [230], but unlike the LSUV initialization, it provides an analytic solution for ELU and MPELU and therefore it has lower computational costs [437]. It was also shown that the MPELU works better with batch normalization compared to the vanilla ELU [437]. The performance of the MPELU was empirically shown on the CIFAR-10 and CIFAR-100 datasets [158] using multiple neural network architectures [437], e.g. nine-layer deep NIN [439] or even a ResNet with 1001 layers [196].

4.2.49 P-E2-ReLU

The AAF named P-E2-ReLU is combining two ELUs and a ReLU using two adaptive parameters [440]. It is defined as

$$f(z_i) = a_i \cdot \text{ReLU}(z_i) + b_i \cdot \text{ELU}(z_i) + (1 - a_i - b_i) \cdot (-\text{ELU}(-z_i)), \quad (315)$$

where a_i and b_i are trainable parameters for each neuron i [440]. The parameters were initialized to $a_i = 0.4$ and $b_i = 0.3$ in experiments in [440]. Jie et al. mentioned that other combinations could be considered and called this family P-E2-XU. One such combination is denoted P-E2-Id and is defined as

$$f(z_i) = a_i z_i + (1 - a_i) \cdot (\text{ELU}(z_i) - \text{ELU}(-z_i)), \quad (316)$$

and another is named P-E2-ReLU-1

$$f(z_i) = a_i \cdot \text{ReLU}(z_i) + (1 - a_i) \cdot (\text{ELU}(z_i) - \text{ELU}(-z_i)), \quad (317)$$

where a_i is a trainable parameter in both AAFs [440]. The parameter was initialized to $a_i = 0.5$ in experiments in [440].

4.2.50 Soft exponential

The soft exponential activation function is an adaptive activation function that is able to interpolate between logarithmic, linear, and exponential functions [441]. It is defined as

$$f(z_i) = \begin{cases} \frac{\exp(z_i) - 1}{a_i} + a_i, & a_i > 0, \\ z_i, & a_i = 0, \\ -\frac{\ln(1 - a_i(z_i + a_i))}{a_i}, & a_i < 0, \end{cases} \quad (318)$$

where a_i is a trainable parameter [441]. The soft exponential activation functions is continuously differentiable with respect to z_i and also with respect to a_i [441]; furthermore, for any constant a_i , the function is monotonic [441]. When $a_i = -1$, the function becomes $f(z_i) = \ln(z_i)$, while for $a_i = 0$ it becomes linear function $f(z_i) = z_i$ and for $a_i = 1$, it is the exponential function $f(z_i) = \exp(z_i)$ [441].

⁵⁶The initialization method was unnamed in the original paper [233] but was later named *Microsoft Research Asia* (MSRA) filler [438].

4.2.51 Continuously differentiable ELU (CELU)

A CELU was proposed in [442]; CELU is very similar to the original parameterization of ELU [1] but reformulated such that the derivative at $z_i = 0$ is 1 for all values of a_i [442]. CELU is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i \left(\exp\left(\frac{z_i}{a_i}\right) - 1 \right), & z_i < 0, \end{cases} \quad (319)$$

where a_i is a learnable parameter for each neuron i . Its main advantages are that its derivative with respect to z_i is bounded and that it contains both the linear transfer function and ReLU [442].

4.2.52 Erf-based ReLU (ErfReLU)

The Erf-based ReLU (ErfReLU) [443] is an AAF similar to the ELU. It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i \operatorname{erf}(z_i), & z_i < 0, \end{cases} \quad (320)$$

where a_i is a learnable parameter for each neuron i and $\operatorname{erf}(z_i)$ is the Gauss error function [443].

4.2.53 Parametric scaled exponential linear unit (PSELU)

A parametric scaled exponential linear unit (PSELU) [444] is basically a SELU (see section 3.7.11) where the parameters a and b controlling the behavior are trainable. It is defined as

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ a_i b_i (\exp(z_i) - 1), & z_i < 0, \end{cases} \quad (321)$$

where a_i and b_i are trainable parameters for each neuron i .

4.2.54 Leaky parametric scaled exponential linear unit (LPSELU)

A leaky parametric scaled exponential linear unit (LPSELU) [444] is a leaky extension of the PSELU (see section 4.2.53) to avoid small gradients hindering the learning process [444]:

$$f(z_i) = \begin{cases} a_i z_i, & z_i \geq 0, \\ a_i b_i (\exp(z_i) - 1) + c_i z_i, & z_i < 0, \end{cases} \quad (322)$$

where a_i and b_i are trainable parameters for each neuron i and c_i is either a predefined constant or a trainable parameter [444].

4.2.55 Leaky parametric scaled exponential linear unit with reposition parameter (LPSELU_RP)

The LPSELU can be extended by a reposition parameter similarly as FReLU extends ReLU (see section 4.2.15) [444]; such function is called LPSELU_RP and is defined as

$$f(z_i) = \begin{cases} a_i z_i + m_i, & z_i \geq 0, \\ a_i b_i (\exp(z_i) - 1) + c_i z_i + m_i, & z_i < 0, \end{cases} \quad (323)$$

where a_i and b_i are trainable parameters for each neuron i , and c_i is either a predefined constant or a trainable parameter the same as for LPSELU (see section 4.2.54) and m_i is a trainable reposition parameter [444]. It was empirically observed that the shift parameter m_i converges to a small negative value, which supports the hypothesis that the negative output of activation functions is important [444].

4.2.56 Shifted ELU family

A family of several activation functions, shifted exponential linear unit, was proposed in [445]; functions in this family have either vertical or horizontal shift of an ELU activation function that can be either constant or trainable. An ELU with fixed horizontal shift is ShELU, with fixed vertical SvELU and PELU (see section 4.2.43) with trainable horizontal shift is PShELU [445]. The ShELU is defined as

$$f(z) = \begin{cases} z + b, & z + b \geq 0, \\ a (\exp(z + b) - 1), & z + b < 0, \end{cases} \quad (324)$$

where a is a fixed parameter similarly as in the vanilla ELU and b is novel, preset parameter controlling the horizontal shift [445]. The SvELU is defined similarly:

$$f(z) = \begin{cases} z + b, & z \geq 0, \\ a(\exp(z) - 1) + b, & z < 0, \end{cases} \quad (325)$$

where a is a fixed parameter similarly as in the vanilla ELU and b is novel, preset parameter controlling the vertical shift [445]. Grelsson and Felsberg define also a variant of PELU with horizontal shift called PSheLU:

$$f(z_i) = \begin{cases} \frac{a_i}{b_i}(z_i + c_i), & z_i + c_i \geq 0, \\ a_i \left(\exp\left(\frac{z_i + c_i}{b_i}\right) - 1 \right), & z_i + c_i < 0, \end{cases} \quad (326)$$

where a_i and b_i are trainable parameters of the original PELU, and c_i is a novel trainable parameter controlling the horizontal shift for each neuron i [445]. For some reason, Grelsson and Felsberg did not propose a PELU with vertical shift (PSvELU), but it could be defined in a similar manner

$$f(z_i) = \begin{cases} \frac{a_i}{b_i}z_i + c_i, & z_i \geq 0, \\ a_i \left(\exp\left(\frac{z_i}{b_i}\right) - 1 \right) + c_i, & z_i < 0, \end{cases} \quad (327)$$

where a_i , b_i are trainable parameters of the original PELU and c_i is a novel trainable parameter controlling the vertical shift. Note that the shifted activation functions with horizontal shifts are equivalent to non-shifted variants with biases that are individual for each neuron and not shared in the same tiling pattern as the convolutional kernel [445].

4.2.57 Tunable swish (T-swish)

The tunable swish (T-swish) proposed in [446] is an AAF combining the ELU, E-swish (see section 4.4.4) and swish (see section 4.4.1) as it has trainable parameters for both horizontal and vertical scaling for negative inputs. It is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq c_i, \\ a_i z_i \cdot \sigma(b_i z_i), & z_i < c_i, \end{cases} \quad (328)$$

where a_i , b_i , and c_i are either fixed or trainable parameters for each neuron i [446].

4.2.58 Rectified parametric sigmoid unit (RePSU)

The rectified parametric sigmoid unit (RePSU) is an AAF proposed in [447]; it consists of a linear combination of two components — rectified parametric sigmoid shrinkage unit (RePSKU) and rectified parametric sigmoid stretchage unit (RePSHU). It is defined as

$$f(z_i) = a_i \text{RePSKU}_{b_i, c_i, d_i, e_i}(z_i) + (1 - a_i) \text{RePSHU}_{b_i, c_i, d_i, e_i}(z_i), \quad (329)$$

where

$$\text{RePSKU}_{b_i, c_i, d_i, e_i}(z_i) = \begin{cases} \frac{z_i - b_i}{1 + \exp\left(-\text{sgn}(z_i - c_i) \left(\frac{|z_i - c_i|}{d_i}\right)^{e_i}\right)}, & z_i \geq b_i, \\ 0, & z_i < b_i, \end{cases} \quad (330)$$

$$\text{RePSHU}_{b_i, c_i, d_i, e_i}(z_i) = \begin{cases} 2z_i - \text{RePSKU}_{b_i, c_i, d_i, e_i}(z_i) & z_i \geq b_i, \\ 0, & z_i < b_i, \end{cases} \quad (331)$$

and a_i , b_i , c_i , d_i , and e_i are parameters (common for both $\text{RePSKU}_{b_i, c_i, d_i, e_i}(z_i)$ and $\text{RePSHU}_{b_i, c_i, d_i, e_i}(z_i)$) [447]. The RePSU is a generalization of the smooth sigmoid-based shrinkage (SSBS) function [448] used for image denoising [447].

4.2.59 Parametric deformable exponential linear unit (PDELU)

An adaptive activation function parametric deformable exponential linear unit (PDELU) [449] is based on the premise that shifting the mean value of the output closer to zero speeds up the learning [449]. The PDELU is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq 0, \\ a_i \left([1 + (1 - b) z_i]^{\frac{1}{1-b}} - 1 \right), & z_i < 0, \end{cases} \quad (332)$$

where a_i is a trainable parameter for each neuron i and b is a fixed hyperparameter controlling the degree of deformation [449]. The Cheng et al. recommend setting $b = 0.9$ [449]. The authors found that the MSRA initialization method [233] is consistent with PDELU [449]. The performance of PDELU was empirically shown on the CIFAR-10 and CIFAR-100 datasets [158] and on the ImageNet dataset [173] where it outperformed ReLU, APLU, LReLU, PReLU, SReLU, ELU, MPELU (and other) activation functions [449].

4.2.60 Elastic exponential linear unit (EELU)

An adaptive variant of the ELU function that has a stochastic component was proposed in [393] — the elastic exponential linear unit (EELU). The EELU combines EReLU (see section 3.6.38) and MPELU (see section 4.2.48) and is defined as

$$f(z_i^c) = \begin{cases} k_i^c z_i^c, & z_i^c \geq 0, \\ a^c (\exp(b^c z_i^c) - 1), & z_i^c < 0, \end{cases} \quad (333)$$

where a^c and b^c are trainable parameters shared among all neurons of a channel c and k_i^c is a randomly sampled noise parameter for each neuron i in channel c during the training stage [393] and set to 1 during the testing stage. The k_i^c is sampled coefficient from Gaussian distribution with a random standard deviation that is truncated from 0 to 2; k_i^c is therefore sampled as

$$k_i^c = \max(0, \min(s_i^c, 2)) \quad (334)$$

where

$$s_i^c \sim N(1, \sigma^2), \quad (335)$$

$$\sigma \sim U(0, \epsilon), \epsilon \in (0, 1], \quad (336)$$

where $N(1, \sigma^2)$ is Gaussian distribution with mean 1 and variance σ^2 , U denotes the uniform distribution [393]. The ϵ is a hyperparameter; the authors recommend smaller values, e.g., 0.1 or 0.2 [393].

The training algorithm is also modified and works in two steps — first, the EELU parameter and the weights are updated with fixed $k_i^c = 1$, and then weights are updated with random k_i^c and fixed EELU parameters [393]. The authors also recommend using the MPELU initialization [437] method [393].

4.2.61 Parametric first power linear unit with sign (PFPLUS)

The parametric first power linear unit with sign (PFPLUS) is an AAF proposed in [304, 450]. It is defined as

$$f(z_i) = a_i z_i \cdot (1 - b_i z_i)^{H(z_i)-1}, \quad (337)$$

where $H(z_i)$ is Heaviside step function (see section 3.1)

$$H(z_i) = \begin{cases} 1, & z_i \geq 0, \\ 0, & z_i < 0. \end{cases} \quad (338)$$

and $a_i > 0$ and $b_i > 0$ are trainable parameters for each neuron i [304]. For example, the PFPLUS is similar to the ReLU when $a_i = 0.2$ and $b_i = 10$ and similar to a linear mapping when $a_i = 5$ and $b_i = 0.1$ [304].

4.2.62 Parametric variational linear unit (PVLU)

The parametric variational linear unit (PVLU) is an adaptive variant of the VLU proposed [243]. It is defined as

$$f(z_i) = \text{ReLU}(z_i) + a_i \sin(b_i z_i) = \max(0, z_i) + a_i \sin(b_i z_i), \quad (339)$$

where a_i and b_i are trainable parameters [243].

4.3 Sigmoid-based adaptive functions

Many different adaptive activation functions based on the sigmoid family were proposed in the literature [1], one of the earliest examples is a logistic sigmoid activation function with shape autotuning [451]. The function proposed by Yamada and Yabuta uses a single parameter controlling both the amplitude and the slope of the activation function [56, 451]. The proposed adaptive function is defined as

$$f(z) = 2 \frac{1 - \exp(-az)}{a(1 + \exp(-az))}, \quad (340)$$

where $a \in (0, \infty)$ is a learnable parameter [56].

4.3.1 Generalized hyperbolic tangent

The generalized hyperbolic tangent [452] introduces two trainable parameters that control the scale of the activation function:

$$f(z_i) = \frac{a_i (1 - \exp(-b_i z_i))}{1 + \exp(-b_i z_i)}, \quad (341)$$

where a_i and b_i are trainable parameters for each neuron i [451]. A non-adaptive version with fixed parameters was used for document recognition in [149] in order to improve convergence toward the end of the learning session [149] (see section 3.2.3).

4.3.2 Trainable amplitude

A more general approach was introduced in [453], which used networks with a trainable amplitude of activation functions; the same approach was later used for recurrent neural networks [454]. The class of adaptive functions with a trainable amplitude is defined as

$$f(z_i) = a_i g(z_i) + b_i, \quad (342)$$

where a_i and b_i are trainable parameters for each neuron i . The a_i determines the trainable amplitude and the b_i trainable offset. These parameters can be either different for each neuron or may be shared by a whole layer or even a whole network [453].

4.3.3 Adaptive slope sigmoidal function (ASSF)

A adaptive slope sigmoidal function (ASSF) based on the work of Yamada and Yabuta, Yamada and Yabuta was used in [456, 457]. It is defined as

$$f(z) = \sigma(a \cdot z), \quad (343)$$

where σ is the logistic sigmoid and a is a global trainable parameter [457]. The ASSF was also rediscovered by Mercioni, Tiron, and Holban in [458].

4.3.4 Slope varying activation function (SVAF)

A slope varying activation function (SVAF) was proposed in [459]

$$f(z) = \tanh(a \cdot z), \quad (344)$$

where a is a global trainable parameter. The slope varying activation function was proposed together with a BP modification that has two different learning rates [459]. The slope varying activation function was implemented as a modification of the BP algorithm rather; a different example of modification of the BP algorithm resulting in an adaptive activation function is presented in [460].

4.3.5 TanhSoft

The TanhSoft is a family of AAFs proposed in [461] that combine the softplus and tanh that contains three notable cases — TanhSoft-1, TanhSoft-2, and TanhSoft-3 [461, 462].

The general TanhSoft is defined as

$$f(z_i) = \tanh(a_i z_i + b_i \exp(c_i z_i)) \ln(d_i + \exp(z_i)), \quad (345)$$

where a_i , b_i , c_i , and d_i are either trainable or fixed parameters [461]; $a_i \in (-\infty, 1]$, $b_i \in [0, \infty)$, $c_i \in (0, \infty)$, and $d_i \in [0, 1]$ [461].

The first AF, named TanhSoft-1, is defined as

$$f(z_i) = \tanh(a_i z_i) \ln(1 + \exp(z_i)), \quad (346)$$

where a_i is a trainable parameter [461, 462]; it can be obtained from the general TanhSoft by setting $b_i = 0$ and $d_i = 1$ [461]. The second AF from [462], TanhSoft-2, is defined as

$$f(z_i) = z_i \tanh(b_i \exp(c_i z_i)), \quad (347)$$

where b_i and c_i are trainable parameters [461, 462]. The TanhSoft-2 can be obtained from the general TanhSoft by setting $a_i = 0$ and $d_i = 0$ [461]. The last AF from [462], TanhSoft-3, is defined as

$$f(z_i) = \ln(1 + \exp(z_i) \tanh(a_i z_i)), \quad (348)$$

where a_i is a trainable parameter [462]. It can be obtained from the general TanhSoft by setting $b_i = 0$ and $d_i = 1$.

4.3.6 Parametric sigmoid (psigmoid)

An adaptive variant of logistic sigmoid named parametric sigmoid (psigmoid)⁵⁷ was proposed in [463, 464].⁵⁸ Similarly as in generalized hyperbolic tangent, it introduces two scaling parameters to a logistic sigmoid:

$$f(z_i) = a_i \sigma(b \cdot z_i), \quad (349)$$

where a_i is a trainable parameter for each neuron or channel i and b is a global trainable parameter [463].

4.3.7 Parametric sigmoid function (PSF)

A parametric sigmoid function (PSF) is a continuous, differentiable, and bounded function proposed in [465, 466]⁵⁹ and is defined as

$$\text{PSF}(z) = \frac{1}{(1 + \exp(-z))^m}, \quad (350)$$

where m is a global trainable parameter [1, 467]. The parameter m controls the slope of the sigmoid and the position of the maximum derivative; the envelope of the relevant derivatives for different values of m is also a sigmoid function [465]. The larger values of m improve the gradient flow [1]. The PSF is only one instance of a larger class of activation functions proposed in [468].

4.3.8 Slope and threshold adaptive activation function with tanh function (STAC-tanh)

The slope and threshold adaptive activation function with tanh function (STAC-tanh) was proposed in [469]. It is basically a tanh based equivalent of the improved logistic sigmoid with adaptive parameters. It is defined as

$$f(z_i) = \begin{cases} \tanh -a_i + b_i (z_i + a_i), & z_i < -a_i, \\ \tanh z_i, & -a_i \leq z_i \leq a_i, \\ \tanh a_i + b_i (z_i - a_i), & z_i > a_i, \end{cases} \quad (351)$$

where a_i and b_i are trainable parameters [469].

4.3.9 Generalized Riccati activation (GRA)

The generalized Riccati activation (GRA) is an adaptive variant of a sigmoid AF proposed in [470]. It is defined as

$$f(z_i) = 1 - \frac{a_i}{a_i + (1 + b_i z_i)^{c_i}}, \quad (352)$$

where a_i , b_i , and c_i are adaptive parameters — $b_i > 0$ and $c_i > 0$ [470].

4.4 Adaptive sigmoid-weighted linear units

There are several AFs that are based on the SiLU but have an adaptive parameter; the most common example is the swish AF, but there are also other popular functions based on the same principle.

4.4.1 Swish

A swish activation function [49] is an adaptive variant of the SiLU [171] (see section 3.3); it is also the member of the LAAF class (see section 4.16):

$$f(z_i) = z_i \cdot \sigma(a_i z_i), \quad (353)$$

where $\sigma(z)$ is the logistic sigmoid, a_i is either a fixed hyperparameter or a trainable parameter [49]. The swish has an output range of $(-\infty, \infty)$ [1]. The parameter a_i controls the amount of non-linearity the swish activation has [1]. The swish might also be considered a member of the family of activate or not activation functions (ACONs) [471]; it is then named ACON-A. The parametric SiLU (PSiLU) is another name for the swish activation used in [384].

⁵⁷Not to be confused with parametric sigmoid function (PSF) from section 4.3.7.

⁵⁸It seems that this AAF was first proposed in 2010 in [464] and then independently in 2021 in [463].

⁵⁹[466] contains the definition equivalent to $f(z) = \text{PSF}(\frac{z}{2})$.

4.4.2 Adaptive hybrid activation function (AHAF)

A swish variant with vertical scaling was proposed in [472] under the name adaptive hybrid activation function (AHAF). It is defined as

$$f(z_i) = a_i z_i \cdot \sigma(b_i z_i), \quad (354)$$

where a_i and b_i are trainable parameters [472].

4.4.3 Parametric shifted SiLU (PSSiLU)

The parametric shifted SiLU (PSSiLU) is a swish based AAF proposed in [384]. It is defined as

$$f(z_i) = \frac{z_i \cdot (\sigma(a_i z_i) - b_i)}{1 - b_i}, \quad (355)$$

where a_i and b_i are trainable parameters [384].

4.4.4 E-swish

E-swish [473] is an AAF inspired by the swish [49] activation function (see section 4.4.1); the E-swish has a scaling parameter that allows for vertical scaling of the activation function [473]. The name of the activation function is not chosen well as the E-swish is rather extending the SiLU (see section 3.3) and not swish which is its adaptive variant.⁶⁰ The function is defined as

$$f(z) = az \cdot \sigma(z), \quad (356)$$

where $\sigma(z)$ is the logistic sigmoid and a is a preset parameter [473] — however, the parameter a is considered to be trainable in review [1]. Alcaide recommends setting $a \in [1, 2]$ to avoid exploding gradients that are hypothesized to more likely occur for higher values of a [473]. The E-swish was found to outperform the SiLU (called swish in the paper) on the the MNIST [182], CIFAR-10 and CIFAR-100 [158] datasets using the Wide ResNet (WRN) [474] architecture [473].

4.4.5 ACON-B

The ACON family consists of swish AF and several extensions; one is named ACON-B and is defined as

$$f(z_i) = (1 - b_i) z_i \cdot \sigma(a_i (1 - b_i) z_i) + b_i z_i, \quad (357)$$

where a_i and b_i are trainable parameters [471]. The b_i is initialized to 0.25 and a_i to 1.⁶¹

4.4.6 ACON-C

The ACON-C is another member of the ACON family from [471]. It is defined as

$$f(z_i) = (c_i - b_i) z_i \cdot \sigma(a_i (c_i - b_i) z_i) + b_i z_i, \quad (358)$$

where a_i , b_i , and c_i are trainable parameters [471, 475]. Ma et al. used initial values $a_i = 1$, $b_i = 0$, and $c_i = 1$ in [471].

Ma et al. also proposed a general extension to the ACON family named MetaACON which uses a small NN to determine the value of the parameter a_i ; they used the variant ACON-C for the experiments with MetaACON resulting in MetaACON-C⁶² [471]. The MetaACON was used to improve YOLOv7 [476] in [477]. Kan et al. extended the ACON AFs into an AF they named CBAC⁶³ [478]. The ACONs were used, for example, in [471, 475, 477, 479–491]. The 1Dmeta-ACON is a MetaACON extension proposed in [492].

4.4.7 Parameterized self-circulating gating unit (PSGU)

The Parameterized self-circulating gating unit (PSGU) [493] is related to the LiSHT and GTU activation functions as it is basically a LiSHT with gated input with learnable scaling parameter. It is defined as

$$f(z_i) = z_i \cdot \tanh(a_i \sigma(z_i)), \quad (359)$$

⁶⁰Calling the SiLU as swish is quite common in the literature, e.g., exponential swish, generalized swish, and TS-swish.

⁶¹There is no initial value for a_i in ACON-B mentioned explicitly in [471]; however, there is one for its extension ACON-C.

⁶²The implementation of MetaACON-C and other AFs from the ACON family is available at <https://github.com/nmaac/acon>.

⁶³No further description is provided in [478].

where a_i is a learnable parameter and $\sigma(z)$ is the logistic sigmoid function [493]. Li et al. also propose a novel initialization method for NNs with the PSGU AF and show that it is more suitable for the use with PSGU than other common methods [493]. The PSGU is shown to outperform ReLU, mish, swish, PATS and GELU using various NIN and ResNet architectures [493]. The PSGU was also proposed in [186] under the name TSReLU learnable (TSReLU) as the adaptive variant of TSReLU. Mercioni, Tat, and Holban used $a_i = 0.5$ as the initial value [186].

4.4.8 Tangent-bipolar-sigmoid ReLU learnable (TBSReLU)

Similarly as TSReLU is an adaptive variant of TSReLU, the TBSReLU learnable (TBSReLU) [186] is an adaptive variant of TBSReLU. This variant is defined as

$$f(z) = z_i \cdot \tanh \left(a_i \frac{1 - \exp(-z_i)}{1 + \exp(-z_i)} \right). \quad (360)$$

where a_i is a trainable parameter [186]. Mercioni, Tat, and Holban used $a_i = 0.5$ as the initial value [186].

4.4.9 PATS

The AF named PATS⁶⁴ [494] is very similar to PSGU, but it uses arctan and a random scaling parameter instead of the tanh and the adaptive parameter in PSGU. It is defined as

$$f(z_i) = z_i \tan^{-1} (a_i \pi \sigma(z_i)), \quad (361)$$

where $\sigma(z)$ is the logistic sigmoid function and

$$a_i \sim U(l, u), \quad (362)$$

is sampled during training⁶⁵ from the uniform distribution with bounds l and u such that $0 < l < u < 1$ [494]. The authors experimented with fixed, deterministic values of $a_i \in \{\frac{1}{4}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}\}$ — the value $\frac{5}{8}$ led to lowest test error on the CIFAR-10 [158]; they also deemed that suitable values for l and u are $\frac{1}{2}$ and 34 respectively [494]. However, only fixed variant with $a_i = \frac{5}{8}$ was used in the follow-up works such as [128].

4.4.10 Adaptive quadratic linear unit (AQuLU)

The adaptive quadratic linear unit (AQuLU) is an adaptive SiLU variant proposed in [179]; it is defined as

$$f(z_i) = \begin{cases} z_i, & z_i \geq \frac{1-b_i}{a_i}, \\ a_i z_i^2 + b_i z_i, & -\frac{b_i}{a_i} \geq z_i < \frac{1-b_i}{a_i}, \\ 0, & z_i < -\frac{b_i}{a_i}, \end{cases} \quad (363)$$

where a_i and b_i are trainable parameters for each neuron i [179].

4.4.11 Sinu-sigmoidal linear unit (SinLU)

Another adaptive SiLU variant is the sinu-sigmoidal linear unit (SinLU), which adds an adaptive term using the sine function to the linear part of the SiLU [495]. The SinLU is defined as

$$f(z_i) = (z_i + a_i \sin(b_i z_i)) \cdot \sigma(z_i), \quad (364)$$

where $\sigma(z_i)$ is the logistic sigmoid function and a_i and b_i are trainable parameters for each neuron i [495].

4.4.12 ErfAct

An AAF based on the Gauss error function was proposed in [496]. The AAF is named ErfAct and is defined as

$$f(z_i) = z_i \cdot \operatorname{erf}(a_i \exp(b_i z_i)), \quad (365)$$

where a_i and b_i are trainable parameters for each neuron i and $\operatorname{erf}(x)$ is the Gauss error function [496].

⁶⁴Not an abbreviation.

⁶⁵Unfortunately, the author did not specify what happens during the test phase in [494], one can only assume that the expected value is used.

4.4.13 Parametric serf (psurf)

An adaptive version of the serf AF named parametric serf (psurf) was proposed in [496]. It is defined as

$$f(z_i) = z_i \cdot \operatorname{erf}(a_i \ln(1 + \exp(b_i z_i))), \quad (366)$$

where a_i and b_i are trainable parameters for each neuron i and $\operatorname{erf}(x)$ is the Gauss error function [496].

4.4.14 Swim

The swim is an adaptive variant of the PFLU (see section 3.7.7) independently proposed in [497]. It is defined as

$$f(z_i) = z_i \cdot \frac{1}{2} \left(1 + \frac{a_i z_i}{\sqrt{1 + z_i^2}} \right), \quad (367)$$

where a_i is either fixed or trainable parameter for each neuron i [497]. Abdool and Dear used fixed $a_i = 0.5$ in their experiments in [497].

4.5 Tuned softmax (tsoftmax)

A softmax (see section 3.5) variant named tuned softmax (tsoftmax) was proposed in [220]; it is defined as

$$f(z_j) = \frac{\int \exp(z_j)}{\sum_{k=1}^N \int \exp(z_k)} + c, \quad (368)$$

where $f(z_j)$ is the output of a neuron j in a softmax layer consisting of N neurons and c is an adaptive parameter [220].

4.6 Generalized Lehmer softmax (glsoftmax)

The generalized Lehmer softmax (glsoftmax) is a softmax variant proposed in [498]. It is defined as

$$f(z_j) = \frac{\exp(\operatorname{LNORM}(z_j))}{\sum_{k=1}^N \exp(\operatorname{LNORM}(z_k))}, \quad (369)$$

where $\operatorname{LNORM}(z_j)$ is a generalized Lehmer-based Z-score-like normalization with four trainable parameters a_i , b_i , c_i , and d_i defined in [498]:

$$\operatorname{LNORM}(z_i) = \frac{z_i - M_{a_i, b_i}}{\operatorname{GLM}_{c_i, d_i}(\mathbf{z} - M_{a_i, b_i})}, \quad (370)$$

$$M_{a_i, b_i} = \operatorname{GLM}_{a_i, b_i}(\mathbf{z}), \quad (371)$$

$$\operatorname{GLM}_{\alpha, \beta}(\mathbf{x}) = \frac{\ln\left(\frac{\sum_{k=1}^N \alpha^{(\beta+1)x_k}}{\sum_{k=1}^N \alpha^{\beta x_k}}\right)}{\ln(\alpha)}, \quad (372)$$

\mathbf{x} is a vector of elements x_k , $k = 1, \dots, N$ and $\mathbf{z} - M_{a_i, b_i}$ represents a vector with elements $z_k - M_{a_i, b_i}$, $k = 1, \dots, N$ [498].

4.7 Generalized power softmax (gpsoftmax)

The generalized power softmax (gpsoftmax) is another softmax variant proposed in [498]. It is defined as

$$f(z_j) = \frac{\exp(\operatorname{PNORM}(z_j))}{\sum_{k=1}^N \exp(\operatorname{PNORM}(z_k))}, \quad (373)$$

where $\operatorname{PNORM}(z_j)$ is a generalized power-based Z-score-like normalization with four trainable parameters a_i , b_i , c_i , and d_i defined in [498]:

$$\operatorname{PNORM}(z_i) = \frac{z_i - M_{a_i, b_i}}{\operatorname{GPM}_{c_i, d_i}(\mathbf{z} - M_{a_i, b_i})}, \quad (374)$$

$$M_{a_i, b_i} = \operatorname{GPM}_{a_i, b_i}(\mathbf{z}), \quad (375)$$

$$\operatorname{GPM}_{\alpha, \beta}(\mathbf{x}) = \frac{\ln\left(\sum_{k=1}^N \alpha^{\beta x_k}\right) - \ln(N)}{\beta \ln(\alpha)}, \quad (376)$$

\mathbf{x} is a vector of elements x_k , $k = 1, \dots, N$ and $\mathbf{z} - M_{a_i, b_i}$ represents a vector with elements $z_k - M_{a_i, b_i}$, $k = 1, \dots, N$ [498].

4.8 Adaptive radial basis function (ARBF)

The adaptive (ARBF) was used in [499]. It is defined as

$$f(z_i) \exp \left(-\frac{(z_i - a_i)^2}{2b_i^2} \right), \quad (377)$$

where a_i and b_i are adaptive parameters for each neuron i [499]. The parameter a_i controls the center while the parameter b_i controls the width [499].

4.9 Parametric Gaussian error linear unit (PGELU)

The AAF named parametric Gaussian error linear unit (PGELU) was proposed in [500] as the result of noise injection. It is an GELU (see section 3.3.1) adaptive variant defined as

$$f(z_i) = z \cdot \Phi \left(\frac{z}{a} \right), \quad (378)$$

where $\Phi(z)$ is the standard Gaussian CDF and a is a global learnable parameter representing the root mean square (RMS) noise [500].

4.10 Parametric flatted-T swish (PFTS)

A parametric flatted-T swish (PFTS) [501] is an adaptive extension of the FTS (see section 3.6.46); PFTS is identical to FTS except for that the parameter T is adaptive — i.e.:

$$f(z_i) = \text{ReLU}(z_i) \cdot \sigma(z_i) + T_i = \begin{cases} \frac{z_i}{1+\exp(-z_i)} + T_i, & z_i \geq 0, \\ T_i, & z_i < 0, \end{cases} \quad (379)$$

where T_i is a trainable parameter for each neuron i [501]; the parameter T_i is initialized to the value -0.20 [501].

4.11 Parametric flatten-p mish (PFPM)

The parametric flatten-p mish (PFPM) is an AAF proposed in [502]; it is defined as

$$f(z_i) = \begin{cases} z_i \tanh(\ln(1 + \exp(z_i))) + p_i, & z_i \geq 0, \\ p_i, & z_i < 0, \end{cases} \quad (380)$$

where p_i is a trainable parameter [502].

4.12 Gaussian error unit (GEU)

The AAF named Gaussian error unit (GEU) was proposed in [500] as the result of noise injection. It is defined as

$$f(z_i) = \Phi \left(\frac{z}{a} \right), \quad (381)$$

where $\Phi(z)$ is the standard Gaussian CDF and a is a global learnable parameter representing the RMS noise [500]. The GEU multiplied by z becomes the PGELU (see section 4.9).

4.13 Scaled-gamma-tanh activation function (SGT)

The scaled-gamma-tanh (SGT) AF is a piecewise polynomial function proposed in [503]. It is defined as

$$f(z_i) = \begin{cases} az_i^{b_i}, & z_i \geq 0, \\ cz_i^{d_i}, & z_i < 0, \end{cases} \quad (382)$$

where a and c are fixed, predefined parameters and b_i and c_i are trainable parameters for each neuron or filter i [503].

4.14 RSign

An adaptive variant of the sign function was used in [389]. It is called react-sign (RSign) and is defined as

$$f(z_i) = \begin{cases} 1, & z_i \geq a_c, \\ -1, & z_i < a_c, \end{cases} \quad (383)$$

where a_c is an adaptive threshold for each channel [389]. An extension was used in [504], where Ding, Liu, and Zhou used multiple RSign functions for each channel.

4.15 P-SIG-RAMP

An AAF combining the logistic sigmoid and ReLU was proposed in [440] under the name P-SIG-RAMP. The P-SIG-RAMP is defined as

$$f(z_i) = a_i \sigma(z_i) + (1 - a_i) \cdot \begin{cases} 1, & z_i \geq \frac{1}{2b_i}, \\ b_i z_i + \frac{1}{2}, & -\frac{1}{2b_i} < z_i < \frac{1}{2b_i}, \\ 0, & z_i \leq -\frac{1}{2b_i}, \end{cases} \quad (384)$$

where $a_i \in [0, 1]$ and b_i are trainable parameters [440].

4.16 Locally adaptive activation function (LAAF)

A general class of slope varying functions called locally adaptive activation function (LAAF) was proposed in [505, 506]:

$$f(z_i) = g(a_i \cdot z_i), \quad (385)$$

where a_i is a trainable parameter for each neuron i and g is any activation function; Jagtap, Kawaguchi, and Karniadakis used logistic sigmoid, tanh, ReLU, and LReLU as g in their LAAFs in [505]. The corresponding activations are thus given by

$$f(z_i) = \sigma(a_i z_i) = \frac{1}{1 + \exp(-a_i z_i)}, \quad (386)$$

$$f(z_i) = \tanh(a_i z_i) = \frac{\exp(a_i z_i) - \exp(-a_i z_i)}{\exp(a_i z_i) + \exp(-a_i z_i)}, \quad (387)$$

$$f(z_i) = \text{ReLU}(a_i z_i) = \max(0, a_i z_i), \quad (388)$$

and

$$f(z_i) = \text{LReLU}(a_i z_i) = \max(0, a_i z_i) - b \max(0, -a_i z_i), \quad (389)$$

where b is the LReLU leakiness parameter [505]. To accelerate the convergence, Jagtap, Kawaguchi, and Karniadakis add additional fixed parameter to the expression:

$$f(z_i) = g(n a_i z_i), \quad (390)$$

where $n > 1$ is a fixed parameter [505]. It was found that this additional parameter improves both the convergence rate and the solution accuracy [505].

4.16.1 Adaptive slope hyperbolic tangent

A tanh activation function with adaptive slope was used in an multi-layer perceptron (MLP) architecture in [507]. The used activation function is defined as

$$f(z_i) = \tanh(a_i z_i), \quad (391)$$

where a_i is a trainable parameter for each neuron i .

4.16.2 Parametric scaled hyperbolic tangent (PSTanh)

A parametric activation function similar to the swish but based on the tanh function instead of the logistic sigmoid called parametric scaled hyperbolic tangent (PSTanh) was proposed in [53]. It is defined as

$$f(z_i) = z_i \cdot a_i (1 + \tanh(b_i z_i)), \quad (392)$$

where a_i and b_i are trainable parameters for each neuron i [53]. The function is also very similar to the PTELU (see section 4.2.30) as for $z_i > 0$ and $a_i \approx 1$, the output is close to z_i [53] (the exact distance depends on the parameters a_i and b_i).

4.16.3 Scaled sine-hyperbolic function (SSinH)

An AF similar to PSTanh is the scaled sine-hyperbolic function (SSinH) [508]; it is defined as

$$f(z_i) = a_i \sinh(b_i z_i), \quad (393)$$

where a_i and b_i are trainable scaling parameters and \sinh is the hyperbolic sine [508].

4.16.4 Scaled exponential function (SExp)

Husain, Ong, and Bober also proposed scaled exponential function (SExp) along with the SSinH in [508]. It is defined as

$$f(z_i) = a_i (\exp(b_i z_i) - 1), \quad (394)$$

where a_i and b_i are trainable scaling parameters and \sinh is the hyperbolic sine [508].

4.16.5 Logmoid activation unit (LAU)

A learnable LAU was proposed in [509, 510]; which utilise two learnable parameters a_l and b_l for each network layer l

$$f(z_{i,l}) = z \ln(1 + a_l \sigma(b_l \cdot z_{i,l})), \quad (395)$$

where $z_{i,l}$ is the output of the neuron i in layer l without the activation function and σ is the logistic sigmoid [510]. The author used initial values of the parameters $a_l = b_l = 1$ for each network's layer l and trained these parameters together with the rest of the network's weights [510].

4.16.6 Cosinu-sigmoidal linear unit (CosLU)

The cosinu-sigmoidal linear unit (CosLU) is an adaptive activation function proposed in [252] that is based on the logistic sigmoid. It is defined as

$$f(z_i) = (z + a_i \cos(b_i z_i)) \sigma(z_i), \quad (396)$$

where a_i and b_i are trainable parameters for neuron i and $\sigma(z_i)$ is the logistic sigmoid function [252]. The cosine amplitude is controlled by the parameter a_i , whereas its frequency is controlled by the parameter b_i .

4.16.7 Adaptive Gumbel (AGumb)

An activation function adaptive Gumbel (AGumb) is based approach of viewing activation functions as a combination of unbounded and bounded components where the bounded component is based upon a cumulative distribution function of a continuous distribution [511]. While the logistic sigmoid activation is a CDF of the symmetric logistic distribution, the AGumb is based on the Gumbel distribution [511]. It is defined as

$$f(z_i) = 1 - (1 + a_i \cdot \exp(z_i))^{-\frac{1}{a_i}}, \quad (397)$$

where $a_i \in \mathbb{R}^+$ is trainable parameter for each neuron i [511].

4.17 Shape autotuning adaptive activation function (SAAAF)

The shape autotuning adaptive activation function (SAAAF)⁶⁶ is an AAF proposed in [512]. It is defined as

$$f(z_i) = \frac{z_i}{\frac{z_i}{a_i} + \exp(-\frac{b_i}{a_i} z_i)}, \quad (398)$$

where $a_i \geq 0$ and $b_i \geq 0$ are trainable parameters for neuron i and $0 < \frac{b_i}{a_i} < e$ [512].

4.18 Noisy activation functions

Stochastic variants of saturating activation functions such as the logistic sigmoid or hyperbolic tangent were proposed in [395] where an additional noise is injected to the activation function when it operates in the saturation regimes [395]. The noisy activation function is defined as

$$f(z_i) = ah(z_i) + (1 - a)u(z_i) - \text{sgn}(z_i)\text{sgn}(1 - a)c \left(\sigma(p_i(h(z_i) - u(z_i))) \right)^2 \epsilon, \quad (399)$$

where $h(z_i)$ is any saturating activation function such as hard-tanh or hard-sigmoid, $u(z_i)$ is its linearization using first-order Taylor expansion around zero, c is a hyperparameter changing the scale of the standard deviation of the noise, p_i is a trainable parameter adjusting the magnitude of the noise for each neuron i , a is a hyperparameter influencing the mean of the added term, and $\sigma(x)$ is the logistic sigmoid function [395]. ϵ is the added noise; it is defined as $\epsilon = |\xi|$

⁶⁶Zhou et al. named the function as *shape autotuning activation function* but the resulting abbreviation SAAF is already taken by smooth adaptive activation function (see section 4.29). Since the proposed function is an AAF, we term it as such to avoid the abbreviation collision.

if the noise term ξ is sampled from half-normal distribution and as $\epsilon = \xi$ if the noise term ξ is sampled from normal distribution with mean 0 and variance 1 [395].

Gulcehre et al. also experimented with adding noise to the input of the activation function, resulting in an activation function defined as

$$f(z_i) = h(z_i + s(z_i)\epsilon), \quad (400)$$

where $s(z_i)$ is either fixed parameter $s(z_i) = b$ or it is a trainable term

$$s(z_i) = c(\sigma(p_i(h(z_i) - u(z_i))))^2, \quad (401)$$

where the meaning of c , σ , p_i , $h(z_i)$, and $u(z_i)$ is same as in eq. (399) [395].

A similar concept in ReLU settings is the ProbAct activation function (see section 4.2.11).

4.19 Fractional adaptive activation functions

Fractional adaptive activation functions (FAAFs) were proposed in [513–517] as a generalization of several activation functions using the fractional calculus (see [518] for a general introduction to the fractional calculus). Generally, for any activation function $f(z)$, its generalization $g(z)$ using fractional derivatives is defined as the a -th fractional derivative of f :

$$g(z) = D^a f(z), \quad (402)$$

where a can be a learnable⁶⁷ parameter [513]. The FAAFs proposed in [515] were further evaluated in [516].

4.19.1 Fractional ReLU

The fractional ReLU (FracReLU) is defined as

$$f(z_i) = \frac{z_i^{1-a_i}}{\Gamma(2-a_i)}, \quad (403)$$

where $\Gamma(x)$ is the Gamma function and a_i is a trainable parameter [513]. The FracReLU was later independently proposed in [515] under the name FReLU (but this abbreviation is already taken by flexible ReLU).

4.19.2 Fractional softplus

The fractional softplus (FracSoftplus) is using the softplus function to generalize sigmoid-like functions through fractional derivatives [513]. It is defined as

$$f(z_i) = D^{a_i} \ln(1 + \exp(z_i)), \quad (404)$$

which is then computed as

$$f(z_i) = \lim_{h \rightarrow 0} \frac{1}{h^{a_i}} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(a_i + 1) \ln(1 + \exp(z_i - nh))}{\Gamma(n+1)\Gamma(1-n+a_i)}, \quad (405)$$

where a_i is a trainable parameter [513]. Particularly interesting cases are when $a_i = 0$ as it is the softplus function, $a_i = 1$ logistic sigmoid, and $a_i = 2$ which leads to a bell-like shape [513].

4.19.3 Fractional hyperbolic tangent

The fractional tanh (FracTanh) is another fractional generalization proposed in [513]; it is defined as

$$f(z_i) = D^{a_i} \tanh(z_i), \quad (406)$$

which is then computed as

$$f(z_i) = \lim_{h \rightarrow 0} \frac{1}{h^{a_i}} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(a_i + 1) \tanh(z_i - n \cdot h)}{\Gamma(n+1)\Gamma(1-n+a_i)}, \quad (407)$$

where a_i is a trainable parameter [513]. The function becomes the tanh for $a_i = 0$ and the quadratic hyperbolic secant function for $a_i = 1$.

⁶⁷[515] did not specify whether the parameter is trainable but [513] explicitly uses a trainable a .

4.19.4 Fractional adaptive linear unit

The fractional adaptive linear unit (FALU) [514] is yet another AAF based on fractional calculus⁶⁸ It can be seen as the fractional generalization using the a_i -th fractional derivative of the swish function:

$$f(z_i) = D^{a_i} z_i \sigma(b_i z_i), \quad (408)$$

where a_i and b_i are trainable parameters and σ is the logistic sigmoid function [514]. The fractional derivative is then calculated as

$$f(z_i) = \lim_{h \rightarrow 0} \frac{1}{h^{a_i}} \sum_{n=0}^{\infty} (-1)^n \frac{\Gamma(a_i + 1) z_i \sigma(b_i z_i)}{\Gamma(n + 1) \Gamma(1 - n + a_i)}. \quad (409)$$

However, as this calculation is not practical, Zamora-Esquivel, Rhodes, and Nachman use following approximation for $a_i \in [0, 2]$ and $b_i \in [1, 10]$:

$$f(z_i) \approx \begin{cases} g(z_i, b_i) + a_i \sigma(b_i z_i) (1 - g(z_i, b_i)), & a_i \in [0, 1], \\ g(z_i, b_i) + a_i \sigma(b_i z_i) (1 - 2h(z_i, b_i)), & a_i \in (1, 2], \end{cases} \quad (410)$$

where

$$g(z_i, b_i) = z_i \sigma(b_i z_i), \quad (411)$$

$$h(z_i, b_i) = g(z_i, b_i) + \sigma(z_i) (1 - g(z_i, b_i)), \quad (412)$$

and a_i and b_i are the two previously mentioned trainable parameters [514]. The FALU was shown to outperform ReLU, GELU, ELU, SELU, and kernel activation function (KAF) on the MNIST [182], CIFAR-10 [158], ImageNet [172, 173], and Fashion MNIST [314] datasets for several tested architectures [514].

4.19.5 Fractional leaky ReLU (FracLReLU)

The fractional LReLU (FracLReLU) is the fractional variant of the LReLU (see section 3.6.2) proposed in [515]. It is defined using fractional calculus as

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} 0.1 z_i, & z_i < 0, \end{cases} \quad (413)$$

where $a_i \in (0, 1)$ is a fixed parameter [515]. The fractional derivative is then calculated as

$$f(z_i) = \begin{cases} \frac{1}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i \geq 0, \\ \frac{b}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i < 0. \end{cases} \quad (414)$$

4.19.6 Fractional parametric ReLU (FracPReLU)

The fractional PReLU (FracPReLU) is the fractional variant of the PReLU (see section 4.2.1) proposed in [515]. It is defined using fractional calculus as

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} b_i z_i, & z_i < 0, \end{cases} \quad (415)$$

where $a_i \in (0, 1)$ is a fixed parameter and b_i is a trainable parameter [515]. The fractional derivative is then calculated as

$$f(z_i) = \begin{cases} \frac{1}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i \geq 0, \\ \frac{b_i}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i < 0. \end{cases} \quad (416)$$

4.19.7 Fractional ELU (FracELU)

The fractional ELU (FracELU) is the fractional variant of the ELU (see section 3.6.48) proposed in [515]. It is defined using fractional calculus as

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} b (\exp(z_i - 1)), & z_i < 0, \end{cases} \quad (417)$$

where $a_i \in (0, 1)$ and b are fixed parameters [515]. The fractional derivative is then calculated as

$$f(z_i) = \begin{cases} \frac{1}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i \geq 0, \\ b \sum_{k=0}^{\infty} \left(\frac{1}{k!} \cdot \frac{\Gamma(k+1)}{\Gamma(k+1-a_i)} z_i^{k-a_i} \right) - b \frac{1}{\Gamma(1-a_i)} z_i^{-a_i}, & z_i < 0. \end{cases} \quad (418)$$

⁶⁸The FALU was published in [514] without any links to [513] even though it was proposed by the same first author and it uses the same principles.

4.19.8 Fractional SiLU (FracSiLU)

The fractional SiLU (FracSiLU) is the fractional variant of the SiLU (see section 3.3) proposed in [515]. It is defined using fractional calculus as

While Job et al. intended the fractional SiLU (FracSiLU) to be the fractional variant of the SiLU (see section 3.3), they used a wrong definition of the SiLU. Here we present both the FracSiLU from the [515] and the FracSiLU that fit the definition of SiLU — the definition from [515] will be denoted as FracSiLU variant 1 (FracSiLU1) whereas the variant we derived as FracSiLU variant 1 (FracSiLU2). The Job et al. used this definition⁶⁹ of SiLU:

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} z_i \sigma(z_i), & z_i < 0. \end{cases} \quad (419)$$

Then the FracSiLU1 is defined as

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} z_i \sigma(z_i), & z_i < 0, \end{cases} \quad (420)$$

where $\sigma(z_i)$ is the logistic sigmoid [515]. The fractional derivative is then calculated as

$$f(z_i) = \begin{cases} \frac{1}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i \geq 0, \\ \sum_{k=0}^{\infty} \left((-1)^k + \frac{(2^{k+1}-1)B_{k+1}\Gamma(k+2)}{\Gamma(k+2-a_i)(k+1)!} \right) z_i^{k+1-a_i}, & z_i < 0, \end{cases} \quad (421)$$

where B_n is n-th Bernoulli's number [515].

When using the SiLU definition from section 3.3, the FracSiLU2 is then defined as

$$f(z_i) = D^{a_i} z_i \sigma(z_i). \quad (422)$$

Since Job et al. made no assumption about the sign of z_i , the fractional derivative of FracSiLU2 is computed as

$$f(z_i) = \sum_{k=0}^{\infty} \left((-1)^k + \frac{(2^{k+1}-1)B_{k+1}\Gamma(k+2)}{\Gamma(k+2-a_i)(k+1)!} \right) z_i^{k+1-a_i}, \quad (423)$$

where B_n is n-th Bernoulli's number.

4.19.9 Fractional GELU (FracGELU)

Similarly as for FracSiLU, Job et al. intended the fractional GELU (FracGELU) to be the fractional variant of the GELU (see section 3.3.1), but they used a wrong definition of the GELU. Here we present both the FracGELU from the [515] and the FracGELU that fit the definition of GELU — the definition from [515] will be denoted as FracGELU variant 1 (FracGELU1) whereas the variant we derived as FracGELU variant 1 (FracGELU2). The Job et al. used this definition⁷⁰ of GELU:

$$f(z) = \begin{cases} z, & z \geq 0, \\ z \cdot \Phi(z), & z < 0, \end{cases} \quad (424)$$

where $\Phi(z)$ is the standard Gaussian CDF [515]. Then the FracGELU1 is defined as

$$f(z_i) = \begin{cases} D^{a_i} z_i, & z_i \geq 0, \\ D^{a_i} z_i \cdot \Phi(z_i), & z_i < 0. \end{cases} \quad (425)$$

The fractional derivative of FracGELU1 is then calculated as

$$f(z_i) = \begin{cases} \frac{1}{\Gamma(2-a_i)} z_i^{1-a_i}, & z_i \geq 0, \\ 0.5 \frac{z_i^{1-a_i}}{\Gamma(2-a_i)} - \frac{1}{\sqrt{2\pi}} \sum_{k=0}^{\infty} \frac{1}{k!} \left(-\frac{1}{2} \right)^k \frac{z_i^{2(k+1)-a_i}}{2k+1} \frac{\Gamma(2k+3)}{\Gamma(2k+3-a_i)}, & z_i < 0. \end{cases} \quad (426)$$

When using the GELU definition from section 3.3.1, the FracGELU2 is then defined as

$$f(z_i) = D^{a_i} z_i \cdot \Phi(z_i). \quad (427)$$

Since Job et al. made no assumption about the sign of z_i , the fractional derivative of FracGELU2 is computed as

$$f(z_i) = 0.5 \frac{z_i^{1-a_i}}{\Gamma(2-a_i)} - \frac{1}{\sqrt{2\pi}} \sum_{k=0}^{\infty} \frac{1}{k!} \left(-\frac{1}{2} \right)^k \frac{z_i^{2(k+1)-a_i}}{2k+1} \frac{\Gamma(2k+3)}{\Gamma(2k+3-a_i)}. \quad (428)$$

⁶⁹Job et al. referenced [38, 432] for their definition of SiLU; however, the [38] contains the SiLU definition from section 3.3 and [432] does not mention SiLU at all.

⁷⁰Job et al. referenced [38, 432] for their definition of SiLU; however, neither [38] nor [432] contains a definition of GELU.

4.20 Scaled softsign

An activation function called scaled softsign [252] is an adaptive variant of the softsign activation (see section 3.2.13) with variable amplitude. It is defined as

$$f(z_i) = \frac{a_i z_i}{b_i + |z_i|}, \quad (429)$$

where a_i and b_i are trainable parameters for each neuron i [252]. The parameter a_i controls the range of the output while the parameter b_i controls the rate of transition between signs [252].

4.21 Parameterized softplus ($s_{+2}L$)

Parameterized softplus is an adaptive variant of a softplus activation function that allows for vertical shifts [64]. It is defined as

$$f(z_i) = \ln(1 + \exp(z_i)) - a_i, \quad (430)$$

where $a_i \in [0, 1]$ is a trainable parameter for each neuron i [64]. Vargas et al. also proposed a non-adaptive variant with fixed a_i that is denoted as s_{+2} [64].

4.22 Universal activation function (UAF)

The so-called universal activation function (UAF) is a softplus based AAF proposed in [332]. It is defined as

$$f(z_i) = \ln(1 + \exp(a_i(z_i + b_i) + c_i z_i^2)) - \ln(1 + \exp(d_i(z_i - b_i))) + e_i, \quad (431)$$

where a_i , b_i , c_i , d_i , and e_i are trainable parameter for each neuron i [332]. For example, the UAF is able to well approximate the step function, logistic sigmoid, tanh, ReLU, LReLU, and Gaussian function [332].

4.23 Learnable extended activation function (LEAF)

The learnable extended activation function (LEAF) is an AAF proposed in [519] that is able to replace several existing AFs. It is defined as

$$f(z_i) = (a_i z_i + b_i) \sigma(c_i z_i) + d_i, \quad (432)$$

where $\sigma(x)$ is the logistic sigmoid and a_i , b_i , c_i , and d_i are trainable parameters for each neuron i [519]. The table 1 contains a list of AFs that are equivalent to a particular LEAF parameterization.

equiv. AF	a_i	b_i	c_i	d_i
ReLU	1	0	$+\infty$	0
SiLU	1	0	1	0
tanh	0	2	2	-1
logistic sigmoid	0	0	1	0
swish	1	0	a_i	0
AHAF	a_i	0	b_i	0

Table 1: **AF equivalent to LEAF parameterizations**

The list of AFs that have an equivalent LEAF parameterization.

4.24 Generalized ReLU (GReLU)

Theb generalized ReLU (GReLU) is an AF based on the UAF (see section 4.22) [498]. It is defined as

$$f(z_i) = \frac{1}{b_i} \log_{a_i} \left(1 + a_i^{b_i z_i} \right) = \frac{\ln \left(1 + a_i^{b_i z_i} \right)}{b_i \ln(a_i)}, \quad (433)$$

where a_i and b_i are trainable parameters [498].

4.25 Multiquadratic activation function (MAF)

The multiquadratic activation function (MAF) was used in [520, 521]. It is defined as

$$f(z_i) = \sqrt{\|z_i - a_i\|^2 + b_i^2}, \quad (434)$$

where a_i and b_i are trainable parameters [521] a_i is the slope coefficient and b_i is the bias coefficient [521].

4.26 EIS activation functions

The EIS⁷¹ is a family of AAFs proposed in [523] with three notable examples EIS-1, EIS-2, and EIS-3 [522, 523].

The general EIS is defined as

$$f(z_i) = \frac{z_i (\ln(1 + \exp(z_i)))^{a_i}}{\sqrt{b_i + c_i z_i^2 + d_i \exp(-e_i z_i)}}, \quad (435)$$

where a_i, b_i, c_i, d_i , and e_i are either trainable parameters or fixed hyperparameters; $a_i \in [0, 1]$, $b_i \in [0, \infty)$, $c_i \in [0, \infty)$, $d_i \in [0, \infty)$, $e_i \in [0, \infty)$ and b_i, c_i , and d_i cannot be equal to zero at the same time [523].

The EIS-1 is defined as

$$f(z_i) = \frac{z_i \ln(1 + \exp(z_i))}{z_i + d_i \exp(-e_i z_i)}, \quad (436)$$

where d_i and e_i are trainable parameters [522, 523]. It can be obtained from the general EIS by setting $a_i = 1$, $b_i = 0$, $c_i = 1$ [523].

The EIS-2 is defined as

$$f(z_i) = \frac{z_i \ln(1 + \exp(z_i))}{\sqrt{b_i + c_i z_i^2}}, \quad (437)$$

where b_i is a trainable parameter [523]. It can be obtained from the general EIS by setting $a_i = 1$, $d_i = 0$ [523]; however, the EIS-2 from [522] also fixes $c_i = 1$.

And finally, the EIS-3 is defined as

$$f(z_i) = \frac{z_i}{1 + d_i \exp(-e_i z_i)}, \quad (438)$$

where d_i and e_i are trainable parameters [522, 523]; it can be obtained from the general EIS by setting $a_i = 0$, $b_i = 1$, $c_i = 0$ [523].

The EIS family contains the softplus, swish, and ISRU as special cases [523].

4.26.1 Linear combination of parameterized softplus and ELU (ELUs+2L)

A linear combination of parameterized softplus and ELU (ELUs+2L) [64] is an adaptive activation function combining ELUs and parameterized softplus activation functions. It is defined as

$$f(z_i) = b_i \text{ELU}(z_i) + (1 - b_i) \text{s+2L}(z_i), \quad (439)$$

where b_i is a trainable parameter for each neuron i , $\text{ELU}(z_i)$ is the ELU activation function and $\text{s+2L}(z_i)$ is the parameterized softplus activation function [64]. The variant with non-adaptive parameterized softplus is denoted as ELUs+2 [64].

4.27 Global-local neuron (GLN)

The global-local neuron (GLN) is an AAF that is a convex combination of two AAFs proposed in [524]. It is defined as

$$f(z_l) = \sigma(a) \cdot \text{global}(z_l) + (1 - \sigma(a)) \cdot \text{local}(z_l) - b, \quad (440)$$

where a_l and b_l are trainable weights for each layer l and $\text{global}(z_l)$ and $\text{local}(z_l)$ are AAFs capable of identifying the global and local characteristics respectively [524, 525]; the authors used $\text{global}(z_l) = \sin(z_l)$ and $\text{local}(z_l) = \tanh(z_l)$ in [524, 525].

4.28 Neuron-adaptive activation function

A similar approach to trainable amplitude and generalized hyperbolic tangent is the so-called neuron-adaptive activation function (NAF) [526–528], which comprises of a linear combination of two activation functions with scalable amplitude:

$$f(z) = a \exp(-b \cdot (z)^2) + \frac{c}{1 + \exp(-d \cdot z)}, \quad (441)$$

where a, b, c , and d are trainable parameters that are shared by the whole network [526]. The NAF was shown to perform superiorly on a few regression tasks [526].

⁷¹The EIS is a name given by Biswas et al.; it is not an abbreviation.

4.28.1 Scaled logistic sigmoid

A scaling variant of logistic sigmoid called scaled logistic sigmoid was proposed in [529]. The function is defined as

$$f(z_i) = \frac{a_i}{1 + \exp(-b_i z_i)}, \quad (442)$$

where a_i and b_i are trainable parameters for each neuron i [529]. Note that this activation is identical to the second part of the previously proposed NAF (see section 4.28).

A variant combining scaled logistic sigmoid with scaled sine (SLS-SS) was also used in [529]; it has four trainable parameters and is defined as

$$f(z_i) = a_i \cdot \sin(b_i z_i) + \frac{c_i}{1 + \exp(-d_i z_i)}, \quad (443)$$

where a_i , b_i , c_i , and d_i are trainable parameters [529]. This activation function is a special case of another variant of NAF [530]:

$$f(z_i) = a_i \cdot \sin(b_i z_i) + c_i \exp(-d_i \cdot (z_i)^2) + \frac{e_i}{1 + \exp(-f_i z_i)}, \quad (444)$$

where a_i , b_i , c_i , d_i , e_i and f_i are trainable parameters [530].

4.29 Adaptive piece-wise linear unit (APLU)

Another generalization of ReLU is the adaptive piece-wise linear unit (APLU), which uses the sum of hinge-shaped functions as the activation function [531]. An approach extending APLU is smooth adaptive activation function (SAAF) with piece-wise polynomial form and was specifically designed for regression and allows for bias–variance trade-off using a regularization term [378].

APLU is defined as

$$f(z_i) = \max(0, z_i) + \sum_{s=1}^S a_i^s \max(0, -z_i + b_i^s), \quad (445)$$

where S is the number of hinges, i is the number of neurons, and a_i^s , b_i^s , $s \in 1, \dots, S$ are trainable parameters per unit [235]. However, the optimizer might choose very large values of a_i^s and balance them by very small weights, which could lead to numerical instabilities; therefore, an L^2 penalty is added to the parameters a_i^s , b_i^s scaled by 0.001 [531]. Another adaptive piecewise linear function was proposed in [532], where a weighted combination of ReLUs with additional parameters was used.

4.30 Simple piecewise linear and adaptive function with symmetric hinges (SPLASH)

The simple piecewise linear and adaptive function with symmetric hinges (SPLASH) [533] is an approach similar to the APLU. It is defined as

$$f(z_l) = \sum_{s=1}^{\frac{S+1}{2}} a_{l,s}^+ \max(0, z - b_{l,s}) + \sum_{s=1}^{\frac{S+1}{2}} a_{l,s}^- \max(0, -z - b_{l,s}), \quad (446)$$

where S is an odd number, $b_{l,s}$ and $-b_{l,s}$ are hinge parameters and $a_{l,s}^+$ and $a_{l,s}^-$ are scaling parameters for each layer l [533]; these max functions form $S + 1$ continuous line segments with hinges at $b_{l,s}$ and $-b_{l,s}$ [533]. While Tavakoli, Agostinelli, and Baldi tried different values for S , they found that using $S = 7$ usually works well [533].

4.31 Multi-bias activation (MBA)

An approach similar to APLU and paired ReLU (see section 4.2.26) termed multi-bias activation (MBA) [534] uses the same activation but with multiple biases, which allows to learn more complex activations; in this it resembles paired ReLU as one input map leads to several output maps with activation with different biases [534]. The weights that will be given to the output maps in the next layer are similar to the weights in the APLU; however, the MBA is able to provide cross-channel information due to multiple outputs for each activation [534]. The MBA is defined as

$$\mathbf{f}(z_i) = \begin{bmatrix} g(z_i + b_{i,1}) \\ g(z_i + b_{i,2}) \\ \vdots \\ g(z_i + b_{i,k}) \\ \vdots \\ g(z_i + b_{i,K}) \end{bmatrix}, \quad (447)$$

where $b_{i,k}$, $k = 1, 2, \dots, K$ are trainable biases and $g(x)$ is any non-linear activation function [534]; Li, Ouyang, and Wang used ReLU as the activation function $g(x)$ [534].

4.32 Mexican ReLU (MeLU)

A Mexican ReLU (MeLU) is an activation function with a similar approach as the APLU, but it does not need any L^2 penalty [535]. The MeLU is defined as

$$f(z_i) = \text{PReLU}(z_i) + \sum_j^{k-1} a_{i,j} \phi_{b_j c_j}(z_i), \quad (448)$$

where $a_{i,j}$ are trainable parameters for each neuron/filter i , and k is the total number of trainable parameters ($k - 1$ for the sum and one for the PReLU), b_j and c_j are fixed constants that are chosen recursively (more details in [535]); $\phi_{b_j c_j}(z_i)$ is defined as

$$\phi_{b_j c_j}(z_i) = \max(c_j - |z_i - b_j|, 0). \quad (449)$$

Maguolo, Nanni, and Ghidoni used $k = 4$ and $k = 8$ for their experiments; the trainable parameters $a_{i,j}$ were all initialized to zero which helps the training at the early stages by exploiting the properties of the ReLU (e.g., the MeLU is convex for many iterations at the beginning)[535]. The advantage of the MeLU over the APLU is that it needs only half of the parameters while retaining the same representation power when the parameters are jointly optimized with the network's weights and biases [535].

4.32.1 Modified Mexican ReLU (MMeLU)

The modified Mexican ReLU (MMeLU) is an MeLU inspired AF proposed in [536]. It is defined as

$$f(z_i) = a_i \cdot \max(b_i - |z_i - c_i|, 0) + (1 - a_i) \text{ReLU}(z_i), \quad (450)$$

where a_i , b_i , and c_i are adaptive parameters estimated using Bayesian procedure outlined in [536]; $a_i \in [0, 1]$, $b_i \in \mathbb{R}^+$ in, and $c_i \in \mathbb{R}$ [536].

4.32.2 Gaussian ReLU (GaLU)

The Gaussian ReLU (GaLU) is a MeLU-inspired AAF proposed in [537]. It uses the same basic form as MeLU has in eq. (448) but it uses following $\phi_{b_j c_j}(z_i)$:

$$\phi_{b_j c_j}(z_i) = \max(c_j - |z_i - b_j|, 0) + \min(|z_i - b_j - 2c_j| - c_j, 0), \quad (451)$$

where b_j and c_j are similar parameters is in the original MeLU; more details about the parameters is available in [537].

4.32.3 Hard-Swish

The Hard-Swish is an adaptive variant of a scaled Hard sigmoid activation [254]. It is defined as

$$f(z_i) = 2z_i \cdot \max(0, \min(0.2b_i z_i + 0.5, 1)), \quad (452)$$

where b_i is either trainable or fixed parameter [254]. For $b_i \rightarrow \infty$, the Hard-Swish approaches the ReLU [254]. The Hard-Swish outperformed the logistic sigmoid, tanh, ReLU, LReLU, and swish on the MNIST dataset [182] in [254]. The ResNet [196], wide residual network (WRN) [474], and DenseNet [198] with glshardswish outperformed their variants with ReLU and swish on the CIFAR-10 [158] dataset [254].

4.33 S-shaped rectified linear activation unit (SReLU)

A S-shaped ReLU (SReLU) [235] consists of three piecewise linear functions that are controlled by four trainable parameters that are learned jointly with the whole network. The SReLU is able to learn both convex and non-convex functions; in particular, it is able to learn both ReLU and also sigmoidlike functions. It is similar to APLU (see section 4.29), but APLU approximates non-convex functions, and it requires the rightmost linear function to have a unit slope and bias of zero [235]. SReLU is defined as

$$f(z_i) = \begin{cases} t_i^r + a_i^r(z_i - t_i^r), & z_i \geq t_i^r, \\ z_i, & t_i^r > z_i > t_i^l, \\ t_i^l + a_i^l(z_i - t_i^l), & z_i \leq t_i^l, \end{cases} \quad (453)$$

where t_i^r , t_i^l , a_i^r , and a_i^l are trainable parameters for each neuron i (or channel i in case of convolutional neural networks) [235]. The parameters t_i^r and t_i^l determine thresholds of an interval outside which the slope of the linear parts is controlled by parameters a_i^r and a_i^l , respectively. The authors Jin et al. show that the SReLU outperformed the ReLU, LReLU, PReLU, APLU, maxout unit and plain NIN on several visual tasks. The authors also recommend to initialize the parameters of SReLU to $t_i \in \mathbb{R}$, $a_i^r := 1$, $t_i^l := 0$, and $a_i^l \in (0, 1)$ which degenerates the SReLU into a LReLU and then keep these parameter fixed during several initial training epochs [235]. The SReLU can be seen as a more general concept to the later proposed piecewise linear unit (PLU) (see section 4.35) and to the BLReLU [251] (see section 3.6.24).

4.33.1 N-activation

The N-activation is activation very similar to a special case of SReLU⁷² proposed in [538]. The N-activation with trainable parameters a_i , and b_i is defined as

$$f(z_i) = \begin{cases} z_i - 2t_{i,\min}, & z_i < t_{i,\min}, \\ -z_i, & t_{i,\min} \leq z_i \leq t_{i,\max}, \\ z_i - 2t_{i,\max}, & z_i > t_{i,\max}, \end{cases} \quad (454)$$

where

$$t_{i,\min} = \min(a_i, b_i) \quad (455)$$

and

$$t_{i,\max} = \max(a_i, b_i). \quad (456)$$

4.33.2 ALiSA

A special case of SReLU was later proposed under the name adaptive LiSA (ALiSA) in [539]; it can be obtained by setting $t_i^r := 1$ and $t_i^l := 0$:

$$f(z_i) = \begin{cases} a_i^r z_i - a_i^r + 1, & z_i \geq 1, \\ z_i, & t_i^r > z_i > t_i^l, \\ a_i^l z_i, & z_i \leq 0, \end{cases} \quad (457)$$

where a_i^r and a_i^l are adaptive parameters [539]. Its nonadaptive variant is called simply linearized sigmoidal activation (LiSA) and has parameters a_i^r and a_i^l fixed [539].

4.34 Alternated left ReLU (All-ReLU)

The alternated left ReLU (All-ReLU) was proposed in [540] for usage in sparse neural networks. It is inspired by the SReLU [540]. It is defined as

$$f(z_i) = \begin{cases} -az_i, & z_i \leq 0 \text{ and } l\%2 = 0, \\ az_i, & z_i \leq 0 \text{ and } l\%2 = 1, \\ z_i, & z_i > 0, \end{cases} \quad (458)$$

where a is a fixed parameter controlling the slope for negative inputs, l is the number of layers, and $\%$ is the modulo operation [540].

4.35 Piecewise linear unit (PLU)

A PLU [541] resembles two earlier proposed activation functions — the SReLU (see section 4.33) and adaptive piece-wise linear unit (see section 4.29); it can be even seen as a special case of the SReLU.

$$f(z_i) = \max(a_i(z_i + b) - b, \min(a_i(z_i - b) + b, z_i)), \quad (459)$$

where a_i is either a trainable parameter or a predefined constant [1] and b is a predefined constant [541]; a variant with $a = 0.1$ and $b = 1$ was shown in [541]. The advantage of the PLU compared to the SReLU is that it produces an invertible function (which is not always the case for the more general SReLU) [541].

⁷²It would be a special case of SReLU if the thresholds were directly trainable and not determined using the min and max functions.

4.36 Adaptive linear unit (AdaLU)

The adaptive linear unit (AdaLU) [542] is yet another piecewise linear AAF. It is defined as

$$f(z_i) = \begin{cases} c_i(z_i - a_i) + b_i, & (z_i - a_i) > 0 \text{ and } c_i(z_i - a_i) > e_i, \\ d_i(z_i - a_i) + b_i, & (z_i - a_i) \leq 0 \text{ and } d_i(z_i - a_i) > e_i, \\ e_i + b_i, & \text{otherwise,} \end{cases} \quad (460)$$

where a_i, b_i, c_i, d_i , and e_i are trainable parameters for each neuron i [542]. The parameters a_i and b_i control the offsets; c_i and d_i control the slope of each linear part, and e_i is the saturation value [542].

4.37 Trapezoid-shaped activation function (TSAF)

The trapezoid-shaped activation function (TSAF) [543] (ref. from [267]) is an AF consisting of four ReLUs. It is defined as

$$f(z_i) = \frac{1}{c_i} (\text{ReLU}(z_i - a_i + c_i) + \text{ReLU}(z_i - a_i) + \text{ReLU}(z_i + b_i - c_i) - \text{ReLU}(z_i - b_i)), \quad (461)$$

where a_i, b_i , and c_i are parameters⁷³ such that $a_i < b_i$ and $c_i \in (0, 1]$ [267].

4.38 Adaptive Richard's curve weighted activation (ARiA)

Another function motivated by the swish activation function is the Adaptive Richard's curve weighted activation (ARiA) [544], which replaces the logistic sigmoid in the swish by Richard's curve [1]. Richard's curve [545] is a generalization of the logistic sigmoid that is controlled by several hyperparameters. The Richard's curve is defined as [544]:

$$\sigma_R(x) = A + \frac{K - A}{(C + Q \cdot \exp(-Bx))^{\frac{1}{v}}}, \quad (462)$$

where A is the lower asymptote, K is the upper asymptote, C is a constant (typically equal to 1 [544]), $v > 0$ controls the direction of growth and B is the exponential growth rate, Q controls the initial value of the function. The ARiA is defined as

$$f(z) = z \cdot \sigma_R(x), \quad (463)$$

where $\sigma_R(x)$ is the Richard's curve from eq. (462) [544]. As such, the ARiA has five hyperparameters controlling its behavior. To reduce the number of the hyperparameters, Adaptive Richard's curve weighted activation 2 (ARiA2) was also proposed [544] that is defined by only two hyperparameters a and b

$$f(z) = z \cdot (1 + \exp(-bz))^{-a}. \quad (464)$$

The swish activation function is a special case of ARiA with $A = 1$, $K = 0$, $B = 1$, $v = 1$, $C = 1$, and $Q = a_i$, where a_i is the parameter of the swish activation function (see section 4.4.1 for details) [544]. The ARiA2 is a special case of ARiA with $K = 0$, $B = 1$, $v = 1$, $C = \frac{1}{a}$, and $Q = b$, where a and b are the ARiA2 hyperparameters [544]. Patwardhan, Ingallhalikar, and Walambe reached best accuracy on the MNIST [182] dataset with a custom CNN using ARiA2 with $a = 1.5$ and $b = 2$; the best parameters for the DenseNet [198] were $a = 1.75$ and $b = 1$ [544]. While the parameters were fixed in the experiments in [544], they can also be trainable as is in the special case of the swish activation function.

4.39 Modified Weibull function

A modified Weibull function (MWF) is an Weibull-function-based AF proposed in [508]. It is defined as

$$f(z_i) = \left(\frac{z_i}{a_i}\right)^{b_i-1} \exp\left(-\left(\frac{z_i}{c_i}\right)^{d_i}\right), \quad (465)$$

where a_i, b_i, c_i , and d_i are trainable parameters [508]. The parameter b_i determines the location of the peak of the AF [508]. The polynomial term dominates for small input values while the exponential starts to dominate with larger values which reduces the output value as the input value further increases [508].

⁷³Pan et al. do not state whether they are used in trainable or fixed form.

4.40 Sincos

The sincos is another older AF proposed in [360]. It is defined as

$$f(z) = a \cdot \sin(bz) + c \cdot \cos(dz), \quad (466)$$

where a , b , c , and d are adaptive parameters [360].

4.41 Combination of sine and logistic sigmoid (CSS)

The combination of sine and logistic sigmoid (CSS)⁷⁴ is an AAF proposed in [464]. It is defined as

$$f(z) = a \cdot \sin(bz) + c \cdot \sigma(dz), \quad (467)$$

where a , b , c , and d are adaptive parameters [464].

4.42 Catalytic activation function (CatAF)

The catalytic activation function (CatAF) is an AAF that uses sinusoidal mixing of any AF and the identity to produce the final activation [546]. It is defined as

$$f(z_i) = z_i \sin(a_i) + g(z_i) \cos a_i, \quad (468)$$

where a_i is a trainable parameter and $g(z_i)$ is any AF such as the ReLU [546].

4.43 Expcos

An AAF combining an exponential function with the cosine was proposed in [464]. It is called expcos in this work⁷⁵ and is defined as

$$f(z) = \exp(-az^2) \cdot \cos(bz), \quad (469)$$

where a and b are adaptive parameters [464].

4.44 Multi-bin trainable linear unit (MTLU)

The multi-bin trainable linear unit (MTLU) can be seen as a conceptual extension of the SReLU (see section 4.33) into more than three segments [1]:

$$f(z_i) = \begin{cases} a_{i,0}z + b_{i,0}, & z_i \geq c_{i,0}, \\ a_{i,1}z + b_{i,1}, & c_{i,0} < z_i \leq c_{i,1}, \\ \dots & \\ a_{i,k}z + b_{i,k}, & c_{i,k-1} < z_i \leq c_{i,k}, \\ \dots & \\ a_{i,K}z + b_{i,K}, & c_{i,K-1} < z_i, \end{cases} \quad (470)$$

where $a_{i,0}, \dots, a_{i,K}$, and $b_{i,0}, \dots, b_{i,K}$ are trainable parameters for each neuron/filter and K and $c_{i,0}, \dots, c_{i,K-1}$ are predefined hyperparameters [547]. The authors used uniformly distributed anchors $c_{i,0}, \dots, c_{i,K-1}$ [547]. The main disadvantage besides the higher number of additional parameters is the higher number of non-differentiable points [1]. The MTLU was also named continuous piecewise nonlinear activation function (CPN)_m in [548]. The CPN_{mc} is a MTLU variant with continuity constraint proposed in [548].

An AF with the same form as the MTLU with only minor differences was proposed in [549, 550] under the name piecewise linear unit (PWL); Zhu et al. also proposed its 2D extension in [550]. Unlike the MTLU, it uses a uniformly spaced demarcation points $c_{i,k}$ [549]. Another PWLU variant named non-uniform piecewise linear unit (N-PWLU) allows for learnable intervals on which the function is piecewise linear, and also it leverages cumulative definition for efficient learning [551]. Multistability analysis of such piecewise linear AFs is analyzed in [552]. An analysis of a number of regions of piecewise linear NNs is available in [553].

⁷⁴The function was unnamed in [464]; we used this abbreviation to distinguish it from SinSig.

⁷⁵The function was originally unnamed in [464].

4.45 Continuous piecewise nonlinear activation function CPN

A variant of the MTLU named CPN where the $c_{i,k}$ was used in [548]. It is defined as

$$f(z_i) = \begin{cases} a_{i,0}z + b_{i,0} + c_{i,0}g(z_i), & z_i \geq d_{i,0}, \\ a_{i,1}z + b_{i,1} + c_{i,1}g(z_i), & d_{i,0} < z_i \leq d_{i,1}, \\ \dots & \\ a_{i,k}z + b_{i,k} + c_{i,k}g(z_i), & d_{i,k-1} < z_i \leq d_{i,k}, \\ \dots & \\ a_{i,K}z + b_{i,K} + c_{i,K}g(z_i), & d_{i,K-1} < z_i, \end{cases} \quad (471)$$

where $a_{i,0}, \dots, a_{i,K}, b_{i,0}, \dots, b_{i,K}, c_{i,0}, \dots, d_{i,K-1}$ are trainable parameters for each neuron/filter, $g(z_i)$ is a non-linear function such as the logistic sigmoid and K and $c_{i,0}, \dots, d_{i,K-1}$ are predefined hyperparameters [548].

Gao et al. also proposed a variant named CPN_{nl}, which introduces a non-linear term for each small interval and does not enforce the uniform division of the activation space [548]. It is defined as

$$f(z_i) = \max \{p_{i,0}(z), p_{1,0}(z), \dots, p_{i,k}(z), \dots, p_{i,K}(z)\}, \quad (472)$$

where

$$p_k(z) = a_{i,k}z + b_{i,k}\text{SiLU}(z_i) + c_i, \quad k = 0, 1, \dots, K, \quad (473)$$

where K is the number of functions and $a_{i,k}, b_{i,k}$, and $c_{i,k}$ are learnable coefficients for $k = 0, 1, \dots, K$ [548].

4.46 Look-up table unit (LuTU)

A piecewise activation function look-up table unit (LuTU) [554–556] is a learnable activation function that consists of several points defining the function; the values between the points are obtained using either linear interpolation or smoothing with single period cosine mask function [554]. Similar adaptive activation function using linear interpolation was used in [557, 558]. A look-up table of anchor points $\{a_{i,j}, b_{i,j}\}, j = 0, 1, \dots, n$ that are uniformly spaced with step s , $a_{i,j} = a_0 + s \cdot j$, controls the shape of the activation functions. The step s , anchor points a_0 , and n are predetermined hyperparameters, and therefore $a_{i,j}$ are predetermined values for which the output values $b_{i,j}$ are learnable parameters. Linear-interpolation-based function is defined as

$$f(z_i) = \frac{1}{s} (b_{i,j} (a_{i,j+1} - z_i) + b_{i,j+1} (z_i - a_{i,j})), \quad a_{i,j} \leq z_i \leq a_{i,j+1}, \quad (474)$$

where $a_{i,j}$ are hyperparameters defined by the step s and initial point a_0 shared for all points and $b_{i,j}$ are trainable parameters for each neuron i [554]. Wang, Liu, and Foroosh used $a_0 = -12$, step $s = 0.1$ and $n = 240$ to cover the interval $[-12, 12]$ using 241 anchor points for each neuron. Therefore, for any input value between $a_{i,j}$ and $a_{i,j+1}$, the output is linearly interpolated from $b_{i,j}$ and $b_{i,j+1}$ [554]. However, such a definition might lead to unstable gradients [554]; therefore, a variant of LuTU with cosine smoothing was also proposed. The smoothing function is defined as

$$r(x, \tau) = \begin{cases} \frac{1}{2\tau} (1 + \cos(\frac{\pi}{\tau}x)), & -\tau \leq x \leq \tau, \\ 0, & \text{otherwise,} \end{cases} \quad (475)$$

where τ is a hyperparameter controlling the period (2τ) of the cosine function [554]. The smoothed variant of the LuTU is then defined as

$$f(z_i) = \sum_{j=0}^n y_j r(z_i - a_{i,j}, ts), \quad (476)$$

where t is an integer defining the ration between τ and s [554]. The formula in eq. (476) can be further simplified as it is not necessary to sum over all $j \in \{0, 1, \dots, n\}$ as the smoothing function has a truncated input domain, more details in [554].

4.47 Maxout unit

Maxout unit returns the maximum of multiple linear functions per each unit i [559]:

$$f(z_i) = \max_{k \in \{1, \dots, K\}} w_i^k z_i + b_i^k \quad (477)$$

where K is the number of linear functions. The maxout unit can also be used directly on inputs of the neuron as shown in [559] (by replacing $w_i^k z_i$ with $x_i^T w_i^k$ where $x_i \in \mathbb{R}^d$ is the vector of individual inputs to a neuron i and $w \in \mathbb{R}^d$ are trainable weights [559]) but the equation presented here uses only the hidden state for simplicity. The advantage of maxout unit is that it is a universal approximator of a convex function [235, 559]; however, it cannot learn non-convex functions [235] and introduces a high number of additional parameters per neuron [235, 559]. While some works show that maxout unit perform superiorly [559, 560], other experiments show that ReLU, which is a special case of maxout, performs better [561]. Furthermore, since the maxout unit is more complex than regular ReLU, the training is relatively slower [561].

Empirical comparison of the maxout unit with ReLU, LReLU, SELU and tanh is available in [561]; with ReLU, tanh, sigmoid and VReLU in [230].

4.48 Adaptive blending unit (ABU)

An approach mixing several activation functions was described in [562] where ABU was introduced. The ABU is a weighted sum of several predefined activations [562]. It is defined as

$$f(z_l) = \sum_{j=0}^n a_{j,l} g_j(z_l), \quad (478)$$

where $g_j(z_l)$ is an activation function from a pool of n activation functions and $a_{j,l}$ is a weighting parameter that is trained for each layer l and activation function $g_j(z_l)$. The ABU was first proposed as a special case of a general framework called TAF already in 1997 [373]. The blending weights $a_{j,l}$ are initialized to $\frac{1}{n}$ but are then trained alongside the weights of the NN [562]. Sütfield et al. used tanh, ELU, ReLU, swish, and the identity as the pool of activation functions g_j but they admit that no exhaustive search was performed to select this set and that there might be other pools that perform better [562]. This approach was also used in [252] where ReLU, logistic sigmoid, tanh, and softsign activation functions were used.

However, similar approach was already proposed in [554] where Wang, Liu, and Foroosh inspired by the mixture of Gaussian unit (MoGU) (see section 4.48.10) generalized the concept to mixing several different activation functions

$$f(z_i) = \sum_{j=0}^n a_{i,j} g_j(z_i - b_{i,j}), \quad (479)$$

where $g_j(z_i)$ is an activation function from a pool of n activation functions, $a_{i,j}$ is a trainable weighting parameter of the function $g_j(z_i)$ and $b_{i,j}$ is a trainable parameter controlling the vertical shift of the function $g_j(z_i)$ for each neuron i [554]. Furthermore, if $g_j(z_i)$ already contains a way for controlling its scale or shift, the parameters $a_{i,j}$ and $b_{i,j}$ can be discarded [554]. This approach is identical to the ABU from [562] if $b_{i,j} = 0$ and the parameters are shared by all neurons in the same layer and not learned for each neuron separately.

A very similar approach was proposed in [563], where Manessi and Rozza use a linear combination of activation functions from a selected pool as the final activation function. The difference from the ABU is that the weights are constrained such that they sum up to 1 [562, 563]. Manessi and Rozza uses analyses the linear combination of identity, ReLU, and tanh activation functions [563]. Sütfield et al. analyzed the performance of unconstrained ABUs and ABUs with various constraints such as $\sum_{j=0}^n a_{j,l} = 1$, $\sum_{j=0}^n |a_{j,l}| = 1$, and two approaches enforcing $\sum_{j=0}^n a_{j,l} = 1$ and $a_{j,l} > 0$ — clipping of negative values $a_{j,l}$ before normalization and softmax normalization [562]. It was found that the unconstrained ABU works the best on average on the selected tasks; however, some of the constrained variants performed better than the unconstrained ABU for particular tasks [562].

Another variant of ABU (called by Klabjan and Harmon *activation ensemble*) was proposed in [564] — the final activation is a weighted sum of activation functions; the weighting coefficients has to sum-up to 1 (similarly to [563]). However, unlike in the work [563], the individual activation functions are scaled before the weighting to the interval $[0, 1]$ using min–max scaling [564]:

$$h_j(z) = \frac{g_j(z) - \min_k (g_j(z_k))}{\max_k (g_j(z_k)) - \min_k (g_j(z_k)) + \epsilon}, \quad (480)$$

where g_j are individual activation functions, ϵ is a small number and k goes through all training samples in a minibatch. The final output is

$$f(z_i) = \sum_{j=0}^n a_{j,i} h_j(z_i), \quad (481)$$

where $a_{j,i}$ is a weight for each neuron i and activation function j , n is the total number of the individual activation functions in the ABU; the weights $a_{j,i} \in [0, 1]$ are constrained such that

$$\sum_{j=0}^n a_{j,i} = 1. \quad (482)$$

4.48.1 Trainable compound activation function (TCA)

The trained compound activation function (TCA) [565] is an AAF similar to the ABU [562] and especially to its variant with the bias (see eq. (479)) [554]; however, unlike the form from eq. (479) it uses horizontal scaling instead of the vertical. It was defined in [565] as

$$f(z_i) = \frac{1}{k} \sum_{j=1}^k f_j(\exp(a_{i,j}) z_i + b_{i,j}), \quad (483)$$

where k is the number of mixed functions and $a_{i,j}$ and $b_{i,j}$, $j = 1, \dots, k$, are scaling and translation trainable parameters for each neuron i and function j [565]. The TCA was found to improve the performance of restricted Boltzmann machines (RBMs) and deep belief networks (DBNs) [565].

Later, Baggenstoss introduced a TCA also with vertical scaling parameters in [566]. This slightly different variant is denoted as trained compound activation function variant 2 (TCAv2) throughout this work. TCAv2 is defined as

$$f(z_i) = \frac{\sum_{j=1}^k \exp(a_{i,j}) f_j(\exp(b_{i,j}) z_i + c_{i,j})}{\sum_{j=1}^k \exp(a_{i,j})}, \quad (484)$$

where k is the number of mixed functions and $a_{i,j}$, $b_{i,j}$ and $c_{i,j}$, $j = 1, \dots, k$, are scaling and translation trainable parameters for each neuron i and function j [566].

4.48.2 Average of a pool of activation functions (APAF)

An average of a pool of activation functions (APAF) was used in [567]; the output is defined as

$$f(z_i) = \frac{\sum_{j=0}^n a_{j,i} h_j(z_i)}{\sum_{j=0}^n a_{j,i}}. \quad (485)$$

Liao used the ReLU, logistic sigmoid, tanh, and the linear functions as the candidate functions in the pool [567]. This approach was also used in [252].

4.48.3 Gating adaptive blending unit (GABU)

Yet another approach previously proposed employs a gated linear combination of activation functions for each neuron [377] — the variant is called gating adaptive blending unit (GABU) throughout this work. This allows each neuron to choose which activation function (from an existing pool) it may use to minimize the error [377]. A similar method uses just binary indicators instead of the gates [568]. The gating variant of ABU from [377] is defined as

$$f(z_i) = \sum_{j=0}^n \sigma(a_{j,i}) g_j(z_i), \quad (486)$$

where $\sigma(a_{j,i})$ is the logistic sigmoid function acting as gating function and $a_{j,i}$ is a trainable parameter controlling the weight of the activation function g_j for each neuron i .

4.48.4 Deep Kronecker neural networks

The concept of ABUs was further generalized in the framework of Deep Kronecker neural networks (DKNNs) [569], which provides an efficient way of constructing wide networks with adaptive activation functions while keeping the number of parameters low [569]. DKNNs are equivalent to the feed-forward neural networks with an adaptive activation function f defined as

$$f(z_l) = \sum_{j=0}^n a_{l,j} g_j(b_{l,j} z_l), \quad (487)$$

where z_l is a preactivation of a neuron from a layer l , $a_{l,j}$ and $b_{l,j}$ are either trainable or fixed parameters and g_j , $j = 1, \dots, n$ are fixed activation functions [569].

4.48.5 Rowdy activation functions

Rowdy activation functions are a general class of activation functions that is a special case of DKNNs (see section 4.48.4). A rowdy activation function is a DKNNs with any activation function (e.g., ReLU) that is the function g_0 from eq. (487) and n other functions that are defined as

$$g_j(z_l) = c \cdot \sin(jcz_l), \quad (488)$$

or

$$g_j(z_l) = c \cdot \cos(jcz_l), \quad (489)$$

where $c \geq 1$ is a fixed scaling factor and $j = 1, \dots, n$ [569]. The rowdy activation functions introduce highly fluctuating, non-monotonic terms that remove saturation regions from the output of each layer in the network [569] similarly as does the stochastic noise in [395].

4.48.6 Self-learnable activation function (SLAF)

The SLAF [570] can be considered to be a special case of the ABU where the function $g_j(z_i)$ are increasing powers of z_i :

$$f(z_i) = \sum_{j=0}^{k-1} a_{i,j} z_i^j, \quad (490)$$

where $a_{i,j}$ are learnable parameters for each neuron i and k is a hyperparameter defining the number of elements in the polynomial expression [1, 570]. However, since the gradient is proportional to z_i and its powers, Goyal, Goyal, and Lall used mean-variance normalization over the training sample to avoid exploding or vanishing gradients [570]. A similar concept was analyzed in [571], where it was applied to the output neuron only. A similar approach was used independently in [572], where authors used the equivalent of SLAF with $k = 6$. A quadratic variant (i.e., SLAF with $k = 2$) was used in [573].

4.48.7 Chebyshev polynomial-based activation function (ChPAF)

A Chebyshev polynomial-based activation function (ChPAF) was proposed in [574]. The function is defined as

$$f(z) = \sum_{j=0}^k a_j C_j(z), \quad (491)$$

where $a_j, j = 0, \dots, k$ are learnable parameters shared by a whole network, k is a fixed hyperparameter denoting the maximum order of used Chebyshev polynomials, and $C_j(z)$ is a Chebyshev polynomial of order j defined as

$$C_{j+1}(z) = 2zC_j(z) - C_{j-1}(z) \quad (492)$$

with starting values $C_0(z) = 1$ and $C_1(z) = z$ [574]. Deepthi, Vikram, and Venkatappareddy used polynomials of a maximum order of 3 in their experiments [574]. The Chebyshev activation function was found to outperform several activation functions including ReLU, ELU, mish and swish while retaining fast convergence using the CIFAR-10 dataset [158] as shown in experiments [574].

4.48.8 Legendre polynomial-based activation function (LPAF)

A Legendre polynomial-based activation function (LPAF) was used for the study of approximations of several non-linearities in [575]. The activation is a linear combination of Legendre polynomials and is defined as

$$f(z) = \sum_{j=0}^k a_j G_j(z), \quad (493)$$

where $a_j, j = 0, \dots, k$ are learnable parameters shared by a whole network, k is a fixed hyperparameter denoting the maximum order of used Legendre polynomials, and $G_j(z)$ is a Legendre polynomial of order j defined as

$$G_{j+1}(z) = \frac{2k+1}{k+1} z G_k(z) - \frac{k}{k+1} G_{k-1}(z), \quad (494)$$

with starting values $G_0(z) = 1$ and $G_1(z) = z$ [575]. The LPAF was found to outperform ELU, ReLU, LReLU, and softplus on the MNIST [182] and Fashion MNIST [314] datasets [575].

4.48.9 Hermite polynomial-based activation function (HPAF)

The Hermite polynomial-based activation function (HPAF) [576] is an AAF similar to ChPAF and LPAF but it used the Hermite polynomials instead. It is defined as

$$f(z) = \sum_{j=0}^k a_j H_j(z), \quad (495)$$

where a_j is a trainable parameter and $H_j(z)$ is the Hermite polynomial

$$H_j(z) = (-1)^j \exp(z^2) \frac{d^j}{dz^j} (\exp(-z^2)), j > 0 \quad (496)$$

and

$$H_0(z) = 1. \quad (497)$$

4.48.10 Mixture of Gaussian unit (MoGU)

The mixture of Gaussian unit (MoGU) was proposed in [554] as a byproduct of analysis of the behavior of the LuTU unit (see section 4.46) as the shape of learned activation units with the cosine smoothing mostly composed of a few peaks and valleys [554]. The MoGU is defined as

$$f(z_i) = \sum_{j=0}^n \frac{a_{i,j}}{\sqrt{2\pi\sigma_{i,j}^2}} \exp\left(-\frac{(z_i - \mu_{i,j})^2}{2\sigma_{i,j}^2}\right), \quad (498)$$

where $a_{i,j}$, $\sigma_{i,j}$, and $\mu_{i,j}$ are trainable parameters for each neuron i and Gaussian j from the mixture [554]. The parameter $a_{i,j}$ controls the scale, $\sigma_{i,j}$ controls the standard deviation, and $\mu_{i,j}$ controls the mean of the Gaussian j for neuron i [554].

4.48.11 Fourier series activation

The Fourier series activation (FSA) was proposed in [567]. It is defined as

$$f(z_i) = a_i + \sum_{j=1}^r (b_{i,j} \cos(jd_i z_i) + c_{i,j} \sin(jd_i z_i)), \quad (499)$$

where a_i , $b_{i,j}$, $c_{i,j}$, d_i are trainable parameters for each neuron i , and r is a fixed hyperparameter denoting the rank of the Fourier series [567]; Liao used $r = 5$ throughout his experiments.

4.49 Padé activation unit (PAU)

Padé activation units (PAUs) [577] are adaptive activations based on the Padé approximant [578, 579]. The PAU is defined as

$$f(z) = \frac{\sum_{j=0}^m a_j z^j}{1 + \sum_{k=1}^n b_k z^k}, \quad (500)$$

where m and n are hyperparameters denoting the order of the polynomials and a_j , $j = 0, \dots, m$ and b_k , $k = 1, \dots, n$ are trainable parameters that are globally shared by all units [577]. While the Padé approximation could be used to approximate particular activation function, the parameters a_j and b_k are optimized freely with other weights of the neural network [577]. This PAU variant was for reinforcement learning in [580] where Delfosse et al. observed that rational functions might replace some of the residual blocks in ResNets. To avoid numerical instabilities, a *safe PAU* ensures that the polynomial in the denominator cannot be zero [577]; it is defined as

$$f(z) = \frac{\sum_{j=0}^m a_j z^j}{1 + |\sum_{k=1}^n b_k z^k|}. \quad (501)$$

The hyperparameters were set to $m = 5$ and $n = 4$ in experiments in [577]. The notion of using rational functions in activations was further analyzed in [581] where authors used activation function equivalent to eq. (500) with distinct parameters for each layer to learn *rational neural networks*; the safe variant of PAU (eq. (501)) was not used as it results in non-smooth activation function and expensive calculation of gradient during training [581]. Boulle, Nakatsukasa, and Townsend used low degrees $m = 3$ and $n = 2$ in their work [581]; this is in contrast to [582] where rational functions of higher orders were used in a graph neural networks.

4.50 Randomized Padé activation unit (RPAU)

The PAU can be extended similarly as RReLU extends ReLU, resulting in randomized Padé activation unit (RPAU) [577]. Let $\mathbf{C} = \{a_0, \dots, a_m, b_0, \dots, b_n\}$ be coefficients of PAU activation (see section 4.49). Then an additive noise is introduced into each coefficient $c_j \in \mathbf{C}$ during training for every input z_k such that $c_{j,k} = c_j + z_{j,k}$, where $z_{j,k} \sim \mathcal{U}(l_j, u_j)$, $l_j = (1 - a)c_j$ and $u_j = (1 + a)c_j$ [577]. This results in RPAU:

$$f(z_k) = \frac{c_{0,k} + c_{1,k}z_k + c_{2,k}z_k^2 + \dots + c_{m,k}z_k^m}{1 + |c_{m+1,k}z_k + c_{m+2,k}z_k^2 + \dots + c_{m+n,k}z_k^n|}, \quad (502)$$

where z_k is output of a unit for training input k [577].

4.51 Enhanced rational activation (ERA)

The enhanced rational activation (ERA) [583] function is very similar to the original PAU (see section 4.49); however, Trimmel et al. note similarly as Boulle, Nakatsukasa, and Townsend that the safe version of PAU is costly to compute whereas the original PAU has undefined values on poles (values of z where the denominator in PAU is equal to zero). To avoid both the poles and the use of absolute value, a modified rational function without the poles is used [583]. The ERA is defined as

$$f(z) = \frac{P(z)}{Q_C(z)} = \frac{\sum_{j=0}^m a_j z^j}{\epsilon + \prod_{k=1}^n \left((z - c_k)^2 + d_k^2 \right)}, \quad (503)$$

where $a_j, j = 0, \dots, m, c_k$, and $d_k, k = 1, \dots, n$ are trainable parameters for each layer and $\epsilon > 0$ is a small number helping to avoid numerical instabilities when d_k are small [583]. In practice, Trimmel et al. used $\epsilon = 10^{-6}$ [583]. The ERA in eq. (503) can be rewritten using partial fractions, which reduces the number of operations and, therefore, leads to more efficient computation [583]. Trimmel et al. used $m = 5$ and $n = 4$ for their experiments.

4.52 Orthogonal Padé activation unit (OPAU)

The orthogonal Padé activation unit (OPAU) is an extension of the PAU proposed in [584]. It is defined as

$$f(z) = \frac{\sum_{j=0}^m a_j r_j(z)}{1 + \sum_{k=1}^n b_k r_k(x)}, \quad (504)$$

where $a_j, j = 0, \dots, m$ and $b_k, k = 1, \dots, n$ are trainable weights, m and n are fixed parameters, and $r_j(z)$ belongs to a set of orthogonal polynomials [584]. The *sage OPAU* is defined⁷⁶ as

$$f(z) = \frac{\sum_{j=0}^m a_j r_j(z)}{1 + \sum_{k=1}^n |b_k| |r_k(x)|}, \quad (505)$$

with identical parameters as the OPAU from eq. (504) [584]. Biswas, Banerjee, and Pandey used six bases for orthogonal polynomials — Chebyshev polynomials (two variants), Hermite polynomials (also two variants), Laguerre, and Legendre polynomials — as shown in table 2.

4.53 Spline interpolating activation functions

More complex approaches include spline interpolating activation functions (SAFs) [103, 376, 379, 585–596], which facilitate the training of a wide variety of activation functions using interpolation. One common example is the cubic spline interpolation that was used in [379]. The SAFs are controlled by a vector $\mathbf{q} \in \mathbb{R}^k$ of internal parameters called *knots*, which are a sampling of the AF over k representative points [379]. The output is computed using a spline interpolation using the closest knot and its p rightmost neighbors; $p = 3$ results in cubic interpolation [379]. Spline-based activation functions were also used in the ExSpliNet — an interpretable approach combining neural networks and ensembles of probabilistic trees [597]. A set of fixed but highly redundant knots for spline interpolation was used in [588], where the authors then relied on the sparsifying effect of L_1 regularization to nullify the coefficients that are not needed [588]. Spline flexible activation functions were used for sound synthesis in [598]. The usage of splines led to the creation of b-spline-based neural networks, e.g., [599].

Similar to the SAF is the piecewise polynomial activation function (PPAF) [600] that is also defined by a number of points where the function switches from one polynomial to another [600]. López-Rubio et al. used zeroth-order,

⁷⁶Using notation as described in the original article by Biswas, Banerjee, and Pandey [584].

polynomial	definition
Chebyshev (first kind)	$r_0(z) = 1, r_1(z) = z, r_{n+1}(z) = 2zr_n(z) - r_{n-1}(z)$
Chebyshev (second kind)	$r_0(z) = 1, r_1(z) = 2z, r_{n+1}(z) = 2zr_n(z) - r_{n-1}(z)$
Laguerre	$r_0(z) = 1, r_1(z) = 1 - z, r_{n+1}(z) = \frac{(2n+1-z)r_n(z) - nr_{n-1}(z)}{n+1}$
Legendre	$r_n(z) = \sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} (-1)^k \frac{(2n-2k)!}{2^n k! (n-2k)! (n-k)!} z^{n-2k}$
Probabilist's Hermite	$r_n(z) = (-1)^n \exp\left(\frac{z^2}{2}\right) \frac{d^n}{dz^n} \left(\exp\left(-\frac{z^2}{2}\right)\right)$
Physicist's Hermite	$r_n(z) = (-1)^n \exp(z^2) \frac{d^n}{dz^n} (\exp(-z^2))$

Table 2: **Polynomial bases used in OPAU**

List of polynomial bases used in the OPAU taken [584]. The Chebyshev polynomials and the Laguerre polynomial have recurrent definitions; whereas the Legendre and the Hermite polynomials are defined by a single expression.

first-order, and third-order polynomials for the piecewise function [600]; for example, zeroth-order PPAF uses step function and is defined as

$$f(z_i) = \begin{cases} 0, & z_i < q_{i,1}, \\ \frac{k}{m}, & q_{i,k} \leq z_i < q_{i,k+1}, \\ 1, & z_i \geq q_{i,m}, \end{cases} \quad (506)$$

where $m - 1$ is the number of controlling points $q_{i,k}$ of a neuron i , $k \in \{1, 2, \dots, m - 1\}$ [600]. The position of control points is determined using the learning procedure outlined in [600].

If there are no constraints and the AF is limited to linear splines, the AF can be also defined using one hidden layer with ReLUs [594]:

$$f(z_i) = \sum_{k=1}^K a_{i,k} \text{ReLU}(b_{i,k} z_i + c_{i,k}), \quad (507)$$

where $K \in \mathbb{N}$ and $a_{i,k}$, $b_{i,k}$, and $c_{i,k}$ are trainable parameters [594].

4.54 Truncated Gaussian unit (TruG)

A truncated gaussian unit (TruG) [601] is a unit in a probabilistic framework that is able to well approximate sigmoid, tanh, and ReLU. It is controlled by truncation points ξ_1 and ξ_2 and under the probabilistic framework described in [601] is defined as

$$E(h|z, \xi_1, \xi_2) = z + \sigma \frac{\phi\left(\frac{\xi_1 - z}{\sigma}\right) - \phi\left(\frac{\xi_2 - z}{\sigma}\right)}{\Phi\left(\frac{\xi_1 - z}{\sigma}\right) - \Phi\left(\frac{\xi_2 - z}{\sigma}\right)}, \quad (508)$$

where $\phi(x)$ is the probability density function (PDF) of a univariate Gaussian distribution with mean z and variance σ^2 and $\Phi(x)$ its CDF [601]. The truncation points can be either selected manually or tuned with the rest of the weights [601].

4.55 Mollified square root function (MSRF) family

Pan et al. used a smoothing approach on piecewise linear AFs to create a whole new family of AFs in [267]. The approach is based on the mollified square root function (MSRF) method. This smoothing approach was first used in [268] and then in the SquarePlus AF in [602], which inspired Pan et al. in the creation of the MSRF family of AFs.

For example, the absolute value $|x|$ is not differentiable at $x = 0$, but it can be regularized by mollification as

$$|x|_\epsilon = \sqrt{x^2 + \epsilon}, \quad (509)$$

where ϵ is a small positive parameter and $\lim_{\epsilon \rightarrow 0+} |x|_\epsilon = |x|$ [267].

4.55.1 SquarePlus

The SquarePlus [602] is the first AF that used the mollification procedure described in section 4.55 above. It is defined as

$$f(z) = \frac{1}{2} (z + |z|_\epsilon) = \frac{1}{2} \left(z + \sqrt{z^2 + \epsilon} \right). \quad (510)$$

The SquarePlus is very similar to the softplus (see section 3.17) for $\epsilon = 4 (\ln(2))^2$ and they produce identical outputs at $z = 0$ [267].

4.55.2 StepPlus

As the SquarePlus approximates the ReLU, the StepPlus approximates the step function (see section 3.1) similarly as the logistic sigmoid does [267]. It is defined as

$$f(z) = \frac{1}{2} \left(1 + \frac{z}{|z|_\epsilon} \right). \quad (511)$$

The sign function is smoothed into the BipolarPlus AF [267]

$$f(z) = \frac{z}{|z|_\epsilon}. \quad (512)$$

4.55.3 LReLUPlus

A smoothed variant of the LReLU called LReLUPlus is defined as

$$f(z_i) = \frac{1}{2} (z_i + a_i z_i + |(1 - a_i) z_i|_\epsilon), \quad (513)$$

where $|x|_\epsilon$ is the MSRF procedure [267] described in section 4.55 and a_i is a fixed or trainable parameter.

A function equivalent to the LReLUPlus was independently proposed in [391] under the name SMU-1. The only difference was that Biswas et al. used parameter μ that is the square root of ϵ from section 4.55: $\epsilon = \mu^2$.

4.55.4 vReLUPlus

The vReLUPlus [267] is a MSRF smoothed variant of the vReLU (see section 3.6.25); it is defined as

$$f(z) = |z|_\epsilon. \quad (514)$$

4.55.5 SoftshrinkPlus

The smoothed variant of the Softshrink (see section 3.6.23) is named SoftshrinkPlus⁷⁷ [267] and is defined as

$$f(z) = z + \frac{1}{2} \left(\sqrt{(z - a)^2 + \epsilon} - \sqrt{(z + a)^2 + \epsilon} \right), \quad (515)$$

where a is a fixed parameter similar to the original Softshrink's thresholding parameter [267].

4.55.6 PanPlus

The MSRF procedure can be also used to smooth the pan AF (see section 3.6.26) [267]; the resulting PanPlus [267] is defined as

$$f(z) = -a + \frac{1}{2} \left(\sqrt{(z - a)^2 + \epsilon} + \sqrt{(z + a)^2 + \epsilon} \right), \quad (516)$$

where a is a fixed thresholding parameter of the pan function [267].

4.55.7 BReLUPlus

The BReLUPlus [267] is a MSRF smoothed variant of the BReLU (see section 3.6.16) defined as

$$f(z) = \frac{1}{2} (1 + |z|_\epsilon - |z - 1|_\epsilon). \quad (517)$$

⁷⁷Pan et al. named the function STFPlus originally [267].

4.55.8 SReLUPlus

Another smoothed AF is the SReLUPlus which is the smoothed variant of the SReLU (see section 4.33) [267]; it is defined as

$$f(z_i) = a_i z_i + \frac{1}{2} (a_i - 1) (|z_i - t_i|_\epsilon - |z_i + t_i|_\epsilon), \quad (518)$$

where a_i has similar role as in the original SReLU and t_i is a parameter for symmetric variant of SReLU with $t_i = t_i^r = t_i^l$ [267].

4.55.9 HardTanhPlus

Similarly, the smoothed variant of the HardTanh (see section 3.6.18) named HardTanhPlus [267] is defined as

$$f(z) = \frac{1}{2} (|z + 1|_\epsilon - |z - 1|_\epsilon). \quad (519)$$

4.55.10 HardshrinkPlus

The smoothed variant of the Hardshrink (see section 3.6.22) is named HardshrinkPlus⁷⁸ [267]; it is defined as

$$f(z) = z \left(1 + \frac{1}{2} \left(\frac{z - a}{\sqrt{(z - a)^2 + \epsilon}} - \frac{z + a}{\sqrt{(z + a)^2 + \epsilon}} \right) \right), \quad (520)$$

where a is a fixed parameter with a similar function as in the Hardshrink [267].

4.55.11 MeLUPlus

Pan et al. also provided a smoothed variant of the MeLU (see section 4.32); however, the formula written in [267] is not the MeLU AF but rather its single component $\phi_{b_j c_j}(z_i)$. Nevertheless, the full smoothed MeLUPlus can be obtained easily as the combination of the LReLUPlus and the smoothed $\phi_{b_j c_j}^{\text{Plus}}(z_i)$ defined as

$$\phi_{b_j c_j}^{\text{Plus}}(z_i) = \frac{1}{2} \left(c_j - |z_i - b_j|_\epsilon + \sqrt{(c_j - |z_i - b_j|_\epsilon)^2 + \epsilon} \right), \quad (521)$$

where b_j and c_j are the same parameters as in the MeLU.

4.55.12 TSAFPlus

The smoothed variant of the TSAF (see section 4.37) named TSAFPlus [267] is defined as

$$f(z_i) = \frac{1}{c_i} (|z_i - a_i + c_i|_\epsilon + |z_i - a_i| + |z_i + b_i - c_i| - |z_i - b_i|), \quad (522)$$

where a_i , b_i , and c_i have a similar role as in the original TSAF [267].

4.55.13 ELUPlus

Even the ELU (see section 3.6.48) can be mollified into a "smoothed" variant named ELUPlus [267]. The smoothed variant is defined as

$$f(z) = \frac{1}{2} (z + |z|_\epsilon) + \frac{1}{2} \left(\frac{\exp(z) - 1}{a} + \left| \frac{\exp(z) - 1}{a} \right|_\epsilon \right), \quad (523)$$

where a is a fixed parameter⁷⁹ with a similar function as in the ELU [267].

4.55.14 SwishPlus

The mollified variant of the swish (see section 4.4.1) named SwishPlus [267] is defined using the smoothed step function instead of the logistic sigmoid; it is, therefore, defined as

$$f(z) = z \cdot \text{StepPlus}(z) = \frac{1}{2} \left(z + \frac{z^2}{|z|_\epsilon} \right). \quad (524)$$

⁷⁸Pan et al. named the function HTFPlus originally [267].

⁷⁹Pan et al. used variant with inverse parameter $\frac{1}{a}$; we have used the same parameter variant as in the original ELU.

4.55.15 MishPlus

The mollified variant of the swish (see section 3.3.29) named MishPlus [267] is defined using the BipolarPlus and SquarePlus as

$$f(z) = z \cdot \text{BipolarPlus}(\text{BipolarPlus}(z)). \quad (525)$$

4.55.16 LogishPlus

The mollified variant of the logish (see section 3.3.11) named LogishPlus [267] is defined as

$$f(z) = z \cdot \ln(1 + \text{StepPlus}(z)). \quad (526)$$

4.55.17 SoftsignPlus

The mollified variant of the softsign (see section 3.2.13) named SoftsignPlus [267] is defined as

$$f(z) = \frac{z}{1 + |z|_\epsilon}. \quad (527)$$

4.55.18 SignReLUPlus

Pan et al. provide a mollified version for an approximation of the SignReLU⁸⁰ (see section 3.6.32) in [267]. They approximate the SignReLU as

$$\text{SignReLU}(z) = \frac{1}{2}(z + |z|) + \frac{z - |z|}{2|1 - z|_\epsilon}. \quad (528)$$

Using the approximation, they then define the SignReLUPlus as

$$\text{SignReLU}(z) = \frac{1}{2}(z + |z|_\epsilon) + \frac{z - |z|_\epsilon}{2|1 - z|_\epsilon}. \quad (529)$$

4.56 Complex approaches

The network in network (NIN) [439], which uses a micro neural network as an adaptive activation function, represents a different approach. A combination of the NIN and maxout units called maxout-in-network (MIN) was shown to have good performance in [603]. A similar approach is the wide hidden expansion (WHE) layer [604], which is a sparsely connected layer with several activation functions that is used in place of a traditional activation function [604].

Adaptive activation functions called NPF that are learned nonparametrically were proposed in [605] where a Fourier series basis expansion is used for nonparametric estimation. Only one NPF is learned per filter in CNNs while different activation is learned in each neuron of a fully connected layer [605]; the learning is in two stages where the network is first learned with ReLUs in the convolution layers and NPF in all others and only then the network is learned with all activation functions being the NPF [605].

Yet another approach is learning activation functions using hypernetworks [606] resting in *hyperactivations*. The hyperactivation consists of two parts — a shallow feed-forward neural network called activation network and a hypernetwork, which is a type of neural network that produces weights for another network [606]. The hypernetwork is used for the normalization of the activation network. A single hyperactivation is learned for each layer in the neural network [606]. A NN with a combination of more activation functions was used in [607].

The adaptive activation function might also be trained in a semi-supervised manner [608–610].

4.56.1 Variable activation function (VAF)

Similarly to NIN, the variable activation function (VAF) subnetwork approach uses simple activation functions to produce more complex behavior [611]; the activation is replaced by a small subnetwork with one hidden layer with k

⁸⁰Pan et al. call it DLU throughout their work [267].

neurons and only one input and one output neuron [611]. Specifically, VAF is defined as

$$f(z_l) = \sum_{j=1}^k a_{l,j} g(b_{l,j} z_l + c_{l,j}) + a_{l,0}, \quad (530)$$

where $a_{l,0}$, $a_{l,j}$, $b_{l,j}$, and $c_{l,j}$, $j = 1, \dots, k$, are trainable parameters for each layer l and $g(x)$ is an activation function such as tanh or ReLU that were used in experiments with VAF in [611]. The same concept of using subnetwork to learn the activation function was also proposed under the name of *activation function unit* (AFU) in [612].

4.56.2 Flexible activation bag (FAB)

The flexible activation bag (FAB) [613] is an approach similar to NIN and VAF as it uses a subnetwork to learn the AF for each layer l using a pool of K activations $f_k(z_l, \mathbf{a}_{l,k})$. It uses a shallow network with double head with ReLU activation in the first layer; then there are two separate heads [613]. The first head predicts the parameters $\mathbf{a}_{l,k}$ of the individual AFs $f_k(z_l, \mathbf{a}_{l,k})$ in the bag squashed by a sigmoid AF, then the parameters are mapped into a valid range of each of the parameters [613]. The second head is a selective head for selecting an appropriate AF by producing a score $s_{l,k}$ — it can be either discrete or continuous resulting in soft or hard selection [613]. Klopries and Schwung used five selection methods — all of the functions are used ($s_{l,k} = 1$), hard selection, soft selection using logistic sigmoids, softmax selection, and Gumber-Softmax [614] selection [613]. The bag of activations used in the FAB consists of a constant function, linear function, exponential function, step function, ReLU, step function, sine function, tanh, logistic sigmoid, and Gaussian function (see [613] for exact definitions with the adaptive parameters). Then the output of FAB is assembled as

$$f(z_l) = \sum_{k=1}^K s_{l,k} f_k(z_l, \mathbf{a}_{l,k}), \quad (531)$$

where $s_{l,k}$ are the selection scores of f_k ; the $\mathbf{a}_{l,k}$ consists of parameters of the function f_k (the used AFs have from one to three parameters) and the f_k are the individual AFs from the bag of K functions [613].

4.56.3 Dynamic parameter ReLU (DY-ReLU)

The dynamic parameter ReLU (DY-ReLU) (proposed under the name of dynamic ReLU in [400] but that collides with previously proposed DReLUs in [398, 399]) is an activation function whose parameters are input dependent [400]. The concept of DY-ReLU is similar to the WHE in [604] and hyperactivations in [606] as the DY-ReLU is an example of a hyperactivation. The dynamic activation function has two components — hyperfunction that computes parameters for the activation function and the activation function itself [400]. The DY-ReLU piecewise linear function is computed as the maximum of multiple linear functions [1]. It is defined as

$$f(z_i) = \max_{1 \leq k \leq K} (a_{i,k} z_i + b_{i,k}), \quad (532)$$

where K is a hyperparameter and $a_{i,k}$ and $b_{i,k}$ are coefficients determined by the hyper function $\theta(z)$ using all inputs z_i [400]. The hyperfunction $\theta(z)$ is a light-weight neural network [400]. The parameters generated by the hyperfunction $\theta(z)$ can be different for each filter i , or they can be shared in the whole layer [400]. The DY-ReLU can be considered as a dynamic and efficient variant of Maxout (see section 4.47) [400].

4.56.4 Random NNs with trainable activation functions

A very different approach based on adaptive activation functions is presented in [615] where a neural network with random weights is initialized, and the weights are not trained, but the activation functions are trained instead. The activation functions in [615] are polynomial activation functions and are trained separately for each hidden neuron with random weights first; only then the weights of the output layer are estimated [615]. Ertuğrul used five different adaptive variants of activation functions:

$$f(z_i) = \frac{1}{1 + \exp(-a_i z_i - b_i)}, \quad (533)$$

$$f(z_i) = \sin(a_i z_i + b_i), \quad (534)$$

$$f(z_i) = \exp(-a_i ||z_i - b_i||), \quad (535)$$

$$f(z_i) = \begin{cases} 1, & a_i z_i + b_i \leq 0, \\ 0, & \text{otherwise,} \end{cases} \quad (536)$$

and

$$f(z_i) = \sqrt{||z_i - a_i||^2 + b_i^2}, \quad (537)$$

where a_i and b_i are trainable parameters [615].

4.56.5 Kernel activation function (KAF)

A kernel activation function (KAF) [616] is a non-parametric function that uses kernel expansion together with a dictionary to make the activation flexible [1]. The KAF uses a weighted sum of kernel terms:

$$f(z_i) = \sum_{j=1}^D a_{i,j} \kappa(z_i, d_j), \quad (538)$$

where D is a fixed hyperparameter, $a_{i,j}$ are mixing coefficients and $d_j, j = 1, \dots, D$ are called *dictionary elements* and $\kappa(z_i, d_j) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is 1D kernel function — Scardapane et al. consider only $a_{i,j}$ trainable and the dictionary elements d_j are uniformly spaced around zero [616]. This has the advantage that the resulting model is linear in its parameters and, therefore, can efficiently optimized [616].

The kernel function $\kappa(z, d_j)$ used in [616] is the 1D Gaussian kernel defined as

$$\kappa(z, d_j) = \exp\left(-\gamma(z - d_j)^2\right), \quad (539)$$

where $\gamma \in \mathbb{R}$ is a fixed parameter called the *kernel bandwidth* [616]. Scardapane et al. recommend setting the kernel bandwidth to

$$\gamma = \frac{1}{6\Delta^2}, \quad (540)$$

where Δ is the distance between the grid points as adapting γ through back-propagation did not yield any gain in accuracy [616]. The mixing coefficients $a_{i,j}$ can be initialized either randomly from a normal distribution — this provided good diversity for the optimization process [616] — or using kernel ridge regression to approximate an activation function of choice [616]. Scardapane et al. also proposed 2D-KAF that works over all possible pairs of incoming values and uses 2D Gaussian kernel [616].

An extension of the KAF approach was presented in [617] where activation function used was the sum of the KAF and RSigELU (see section 3.7.15) or KAF and RSigELUD (see section 3.7.19). Kernel methods are becoming more common in deep learning — e.g., fully kerneted layers are replacing fully connected layers with a kernel-based approach in [618].

4.57 SAVE-inspired activation functions

Brad produced several AF that are, supposedly, motivated by human behavior in [619]. These AFs were created using the SAVE method [620] and are mostly variations of the AFs listed above. For completeness' sake, a list of these AFs is included in our work in table 3 also with the real-life motivations listed in [619] — however, no deeper analysis or objective evaluation of these AFs was not provided in [619].

formula	parameters	principle from [619]
$a \sin(bz + c)$	a, b, c	activation of resonance
$a \tanh(bz + c)$	a, b, c	activation of resonance
$z + a$	a	introduction of neutral elements
$a (\text{ReLU}(z))^b$	a, b	action against the wolf-pack spirit
$a \exp(bz)$	a, b	activation of centrifugal forces
$a_1 \text{ReLU}(f_1(z)) + \dots + a_n \text{ReLU}(f_n(z))$	$a_1, \dots, a_n,$ $f_1(z), \dots, f_n(z)$	application of multi-level connections
$\text{ReLU}(z - a)$	a	application of asymmetry
$a_1 \text{ReLU}(z) + \dots + a_n \text{ReLU}(z)$	a_1, \dots, a_n	harmonization of individual goals with collective goal
az^{1-b}	a, b	transformation for value-added
$a(1 - \exp(-bz))$	a, b	transformation for value-added
$\text{ReLU}(f(z)g(z))$	$f(z), g(z)$	application of prisoner paradox
$\text{ReLU}\left(\frac{\exp(z)}{1 + \exp(-az)}\right)$	a	application of prisoner paradox
$\text{ReLU}\left(\frac{az+b}{cz+d}\right)$	a, b, c, d	application of shipwrecked paradox

Table 3: **SAVE-inspired activations**
The list of SAVE-inspired AFs from [619].

5 Conclusion

This paper provides an extensive survey of 400 neural network activation functions. Despite all its scope, it has some limitations and focuses on the largest family of real-valued activation functions categorized into two main classes: fixed activation functions and adaptive activation functions. The fixed activation functions that we also refer to as classical are predetermined mathematical functions that apply the same transformation to all inputs regardless of their values. Each neuron within a layer typically applies the same activation function to its inputs. Examples include logistic sigmoid, hyperbolic tangent, and ReLU [221] functions. On the other hand, adaptive activation functions learn their parameters based on the input data and may thus change their shape. This feature allows for more flexibility and leads to faster convergence during training and improved performance [505]. Examples of adaptive activation functions include PReLU [233], PELU [431], swish [49], and TAAF [380, 381].

The profound impact of activation functions on network performance is undeniable, and the absence of a consolidated resource often results in redundant proposals and wasteful reinvention. By offering this comprehensive compilation, we aim to prevent the unnecessary duplication of established activation functions. While recognizing the limitations of our work in not conducting extensive benchmarks or in-depth analyses, we believe this exhaustive list will be a valuable reference for researchers. Even though this list will never be complete due to ongoing proposals of new activation functions, we believe it establishes a solid foundation for future research.

References

- [1] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri. "Activation functions in deep learning: A comprehensive survey and benchmark." In: *Neurocomputing* 503 (Sept. 2022), pp. 92–108. DOI: 10.1016/j.neucom.2022.06.111. URL: <https://doi.org/10.1016/j.neucom.2022.06.111> (cit. on pp. 1–3, 5, 6, 9, 13, 15–18, 20, 22–24, 28, 29, 40, 42, 45–47, 50, 51, 53, 55, 56, 69–71, 75, 82, 83).
- [2] A. Apicella, F. Donnarumma, F. Isgro, and R. Prevete. "A survey on modern trainable activation functions." In: *Neural Networks* 138 (June 2021), pp. 14–32. DOI: 10.1016/j.neunet.2021.01.026. URL: <https://doi.org/10.1016/j.neunet.2021.01.026> (cit. on pp. 1–3).
- [3] M. Çelebi and M. Ceylan. "The New Activation Function for Complex Valued Neural Networks: Complex Swish Function." In: *4th International Symposium on Innovative Approaches in Engineering and Natural Sciences Proceedings*. SETSCI, July 2019. DOI: 10.36287/setsci.4.6.050. URL: <https://doi.org/10.36287/setsci.4.6.050> (cit. on p. 2).
- [4] Y. Zhang, Q. Hua, H. Wang, Z. Ji, and Y. Wang. "Gaussian-type activation function with learnable parameters in complex-valued convolutional neural network and its application for PolSAR classification." In: *Neurocomputing* 518 (Jan. 2023), pp. 95–110. DOI: 10.1016/j.neucom.2022.10.082. URL: <https://doi.org/10.1016/j.neucom.2022.10.082> (cit. on p. 2).
- [5] S. Scardapane, S. V. Vaerenbergh, A. Hussain, and A. Uncini. "Complex-Valued Neural Networks With Nonparametric Activation Functions." In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 4.2 (Apr. 2020), pp. 140–150. DOI: 10.1109/tetci.2018.2872600. URL: <https://doi.org/10.1109/tetci.2018.2872600> (cit. on p. 2).
- [6] B. N. Örnek, S. B. Aydemir, T. Düzenli, and B. Özak. "Some remarks on activation function design in complex extreme learning using Schwarz lemma." In: *Neurocomputing* 492 (July 2022), pp. 23–33. DOI: 10.1016/j.neucom.2022.04.010. URL: <https://doi.org/10.1016/j.neucom.2022.04.010> (cit. on p. 2).
- [7] Q. Hua, Y. Zhang, Y. Jiang, and H. Mu. "Gaussian-type activation function for complex-valued CNN and its application in polar-SAR image classification." In: *Journal of Applied Remote Sensing* 15.02 (May 2021). DOI: 10.1117/1.jrs.15.026510. URL: <https://doi.org/10.1117/1.jrs.15.026510> (cit. on p. 2).
- [8] T. Kim and T. Adali. "Fully Complex Multi-Layer Perceptron Network for Nonlinear Signal Processing." In: *The Journal of VLSI Signal Processing* 32.1/2 (2002), pp. 29–43. DOI: 10.1023/a:1016359216961. URL: <https://doi.org/10.1023/a:1016359216961> (cit. on p. 2).
- [9] R. Savitha, S. Suresh, N. Sundararajan, and P. Saratchandran. "A new learning algorithm with logarithmic performance index for complex-valued neural networks." In: *Neurocomputing* 72.16–18 (Oct. 2009), pp. 3771–3781. DOI: 10.1016/j.neucom.2009.06.004. URL: <https://doi.org/10.1016/j.neucom.2009.06.004> (cit. on p. 2).
- [10] G.-B. Huang, M.-B. Li, L. Chen, and C.-K. Siew. "Incremental extreme learning machine with fully complex hidden nodes." In: *Neurocomputing* 71.4–6 (Jan. 2008), pp. 576–583. DOI: 10.1016/j.neucom.2007.07.025. URL: <https://doi.org/10.1016/j.neucom.2007.07.025> (cit. on p. 2).
- [11] R. Savitha, S. Suresh, N. Sundararajan, and H. Kim. "A fully complex-valued radial basis function classifier for real-valued classification problems." In: *Neurocomputing* 78.1 (Feb. 2012), pp. 104–110. DOI: 10.1016/j.neucom.2011.05.036. URL: <https://doi.org/10.1016/j.neucom.2011.05.036> (cit. on p. 2).
- [12] J. Hu, H. Tan, and C. Zeng. "Global exponential stability of delayed complex-valued neural networks with discontinuous activation functions." In: *Neurocomputing* 416 (Nov. 2020), pp. 1–11. DOI: 10.1016/j.neucom.2020.02.006. URL: <https://doi.org/10.1016/j.neucom.2020.02.006> (cit. on p. 2).
- [13] M. Tan and D. Xu. "Multiple μ -stability analysis for memristor-based complex-valued neural networks with nonmonotonic piecewise nonlinear activation functions and unbounded time-varying delays." In: *Neurocomputing* 275 (Jan. 2018), pp. 2681–2701. DOI: 10.1016/j.neucom.2017.11.047. URL: <https://doi.org/10.1016/j.neucom.2017.11.047> (cit. on p. 2).
- [14] Y. Kuroe, M. Yoshida, and T. Mori. "On Activation Functions for Complex-Valued Neural Networks — Existence of Energy Functions." In: *Artificial Neural Networks and Neural Information Processing — ICANN/ICONIP 2003*. Springer Berlin Heidelberg, 2003, pp. 985–992. DOI: 10.1007/3-540-44989-2_117. URL: https://doi.org/10.1007/3-540-44989-2_117 (cit. on p. 2).
- [15] N. Özdemir, B. B. İskender, and N. Y. Özgür. "Complex valued neural network with Möbius activation function." In: *Communications in Nonlinear Science and Numerical Simulation* 16.12 (Dec. 2011), pp. 4698–4703. DOI: 10.1016/j.cnsns.2011.03.005. URL: <https://doi.org/10.1016/j.cnsns.2011.03.005> (cit. on p. 2).
- [16] J. Gao, B. Deng, Y. Qin, H. Wang, and X. Li. "Enhanced Radar Imaging Using a Complex-Valued Convolutional Neural Network." In: *IEEE Geoscience and Remote Sensing Letters* 16.1 (Jan. 2019), pp. 35–39. ISSN: 1558-0571. DOI: 10.1109/lgrs.2018.2866567. URL: <http://dx.doi.org/10.1109/LGRS.2018.2866567> (cit. on p. 2).
- [17] C. Lee, H. Hasegawa, and S. Gao. "Complex-Valued Neural Networks: A Comprehensive Survey." In: *IEEE/CAA Journal of Automatica Sinica* 9.8 (Aug. 2022), pp. 1406–1426. DOI: 10.1109/jas.2022.105743. URL: <https://doi.org/10.1109/jas.2022.105743> (cit. on p. 2).

- [18] A. D. Jagtap and G. E. Karniadakis. "How Important Are Activation Functions in Regression and Classification? A Survey, Performance Comparison, and Future Directions." In: *Journal of Machine Learning for Modeling and Computing* 4.1 (2023), pp. 21–75. DOI: 10.1615/jmachlearnmodelcomput.2023047367. URL: <https://doi.org/10.1615/jmachlearnmodelcomput.2023047367> (cit. on p. 2).
- [19] N. Vieira. "Bicomplex Neural Networks with Hypergeometric Activation Functions." In: *Advances in Applied Clifford Algebras* 33.2 (Mar. 2023). ISSN: 1661-4909. DOI: 10.1007/s00006-023-01268-w. URL: <http://dx.doi.org/10.1007/s00006-023-01268-w> (cit. on p. 2).
- [20] D. García-Retuerta, R. Casado-Vara, A. Martín-del Rey, F. De la Prieta, J. Prieto, and J. M. Corchado. "Quaternion Neural Networks: State-of-the-Art and Research Challenges." In: *Intelligent Data Engineering and Automated Learning – IDEAL 2020*. Springer International Publishing, 2020, pp. 456–467. ISBN: 9783030623654. DOI: 10.1007/978-3-030-62365-4_43. URL: http://dx.doi.org/10.1007/978-3-030-62365-4_43 (cit. on p. 2).
- [21] X. Zhu, Y. Xu, H. Xu, and C. Chen. "Quaternion Convolutional Neural Networks." In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018. URL: https://openaccess.thecvf.com/content_ECCV_2018/papers/Xu_Yu_Zhu_Quaternion_Convolutional_Neural_Networks_ECCV_2018_paper.pdf (cit. on p. 2).
- [22] C.-A. Popa. "Scaled Conjugate Gradient Learning for Quaternion-Valued Neural Networks." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 243–252. ISBN: 9783319466750. DOI: 10.1007/978-3-319-46675-0_27. URL: http://dx.doi.org/10.1007/978-3-319-46675-0_27 (cit. on p. 2).
- [23] C.-A. Popa. "Learning Algorithms for Quaternion-Valued Neural Networks." In: *Neural Processing Letters* 47.3 (Sept. 2017), pp. 949–973. ISSN: 1573-773X. DOI: 10.1007/s11063-017-9716-1. URL: <http://dx.doi.org/10.1007/s11063-017-9716-1> (cit. on p. 2).
- [24] S. Yu, H. Li, X. Chen, and D. Lin. "Multistability analysis of quaternion-valued neural networks with cosine activation functions." In: *Applied Mathematics and Computation* 445 (May 2023), p. 127849. ISSN: 0096-3003. DOI: 10.1016/j.amc.2023.127849. URL: <http://dx.doi.org/10.1016/j.amc.2023.127849> (cit. on p. 2).
- [25] Z. Xu, B. Tang, X. Zhang, J. F. Leong, J. Pan, S. Hooda, E. Zamburg, and A. V.-Y. Thean. "Reconfigurable nonlinear photonic activation function for photonic neural network based on non-volatile opto-resistive RAM switch." In: *Light: Science & Applications* 11.1 (Oct. 2022). ISSN: 2047-7538. DOI: 10.1038/s41377-022-00976-5. URL: <http://dx.doi.org/10.1038/s41377-022-00976-5> (cit. on p. 2).
- [26] P. V. de Campos Souza. "Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature." In: *Applied Soft Computing* 92 (July 2020), p. 106275. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2020.106275. URL: <http://dx.doi.org/10.1016/j.asoc.2020.106275> (cit. on p. 2).
- [27] R. Das, S. Sen, and U. Maulik. "A Survey on Fuzzy Deep Neural Networks." In: *ACM Computing Surveys* 53.3 (May 2020), pp. 1–25. ISSN: 1557-7341. DOI: 10.1145/3369798. URL: <http://dx.doi.org/10.1145/3369798> (cit. on p. 2).
- [28] S. L. Bangare. "Classification of optimal brain tissue using dynamic region growing and fuzzy min-max neural network in brain magnetic resonance images." In: *Neuroscience Informatics* 2.3 (Sept. 2022), p. 100019. ISSN: 2772-5286. DOI: 10.1016/j.neuri.2021.100019. URL: <http://dx.doi.org/10.1016/j.neuri.2021.100019> (cit. on p. 2).
- [29] M. Malcangi and G. Nano. "Biofeedback: e-health prediction based on evolving fuzzy neural network and wearable technologies." In: *Evolving Systems* 12.3 (Mar. 2021), pp. 645–653. ISSN: 1868-6486. DOI: 10.1007/s12530-021-09374-5. URL: <http://dx.doi.org/10.1007/s12530-021-09374-5> (cit. on p. 2).
- [30] J. Fei, Z. Wang, X. Liang, Z. Feng, and Y. Xue. "Fractional Sliding-Mode Control for Microgyroscope Based on Multilayer Recurrent Fuzzy Neural Network." In: *IEEE Transactions on Fuzzy Systems* 30.6 (June 2022), pp. 1712–1721. ISSN: 1941-0034. DOI: 10.1109/tfuzz.2021.3064704. URL: <http://dx.doi.org/10.1109/tfuzz.2021.3064704> (cit. on p. 2).
- [31] X.-h. Liu, D. Zhang, J. Zhang, T. Zhang, and H. Zhu. "A path planning method based on the particle swarm optimization trained fuzzy neural network algorithm." In: *Cluster Computing* 24.3 (Jan. 2021), pp. 1901–1915. ISSN: 1573-7543. DOI: 10.1007/s10586-021-03235-1. URL: <http://dx.doi.org/10.1007/s10586-021-03235-1> (cit. on p. 2).
- [32] J. A. Duesch, T. A. Catanach, and N. Das. *Adaptive n-ary Activation Functions for Probabilistic Boolean Logic*. 2022. DOI: 10.48550/ARXIV.2203.08977. URL: <https://arxiv.org/abs/2203.08977> (cit. on p. 2).
- [33] L. Parisi, D. Neagu, R. Ma, and F. Campean. "Quantum ReLU activation for Convolutional Neural Networks to improve diagnosis of Parkinson's disease and COVID-19." In: *Expert Systems with Applications* 187 (Jan. 2022), p. 115892. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2021.115892. URL: <http://dx.doi.org/10.1016/j.eswa.2021.115892> (cit. on p. 2).
- [34] A. Farzad, H. Mashayekhi, and H. Hassanpour. "A comparative performance analysis of different activation functions in LSTM networks for classification." In: *Neural Computing and Applications* 31.7 (Oct. 2017), pp. 2507–2521. DOI: 10.1007/s00521-017-3210-6. URL: <https://doi.org/10.1007/s00521-017-3210-6> (cit. on pp. 2, 6).
- [35] G. Bingham and R. Miikkulainen. "Discovering Parametric Activation Functions." In: *Neural Networks* 148 (Apr. 2022), pp. 48–65. DOI: 10.1016/j.neunet.2022.01.001. URL: <https://doi.org/10.1016/j.neunet.2022.01.001> (cit. on p. 2).
- [36] M. Badiger and J. A. Mathew. "Retrospective Review of Activation Functions in Artificial Neural Networks." In: *Proceedings of Third International Conference on Communication, Computing and Electronics Systems*. Springer Singapore, 2022, pp. 905–919. DOI: 10.1007/978-981-16-8862-1_59. URL: https://doi.org/10.1007/978-981-16-8862-1_59 (cit. on p. 2).
- [37] A. D. Jagtap and G. E. Karniadakis. *How important are activation functions in regression and classification? A survey, performance comparison, and future directions*. 2022. DOI: 10.48550/ARXIV.2209.02681. URL: <https://arxiv.org/abs/2209.02681> (cit. on p. 2).
- [38] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. 2018. DOI: 10.48550/ARXIV.1811.03378. URL: <https://arxiv.org/abs/1811.03378> (cit. on pp. 2, 64).
- [39] W. Duch and N. Jankowski. "Taxonomy of neural transfer functions." In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. IEEE, 2000. DOI: 10.1109/ijcnn.2000.861353. URL: <http://dx.doi.org/10.1109/ijcnn.2000.861353> (cit. on p. 2).
- [40] B. Raitani. "Survey on recent activation functions with emphasis on oscillating activation functions." In: (June 2022). DOI: 10.31224/2429. URL: <http://dx.doi.org/10.31224/2429> (cit. on p. 2).
- [41] L. Datta. *A Survey on Activation Functions and their relation with Xavier and He Normal Initialization*. 2020. DOI: 10.48550/ARXIV.2004.06632. URL: <https://arxiv.org/abs/2004.06632> (cit. on p. 2).
- [42] M. A. Mercioni and S. Holban. "A Brief Review of the Most Recent Activation Functions for Neural Networks." In: *2023 17th International Conference on Engineering of Modern Electric Systems (EMES)*. IEEE, June 2023. DOI: 10.1109/emes58375.2023.10171705. URL: <http://dx.doi.org/10.1109/emes58375.2023.10171705> (cit. on p. 2).
- [43] A. D. Rasamoelina, F. Adjailia, and P. Sincak. "A Review of Activation Function for Artificial Neural Network." In: *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMII)*. IEEE, Jan. 2020. DOI: 10.1109/samii48414.2020.9108717. URL: <http://dx.doi.org/10.1109/samii48414.2020.9108717> (cit. on p. 2).
- [44] M. A. Mercioni and S. Holban. "The Most Used Activation Functions: Classic Versus Current." In: *2020 International Conference on Development and Application Systems (DAS)*. IEEE, May 2020. DOI: 10.1109/das49615.2020.9108942. URL: <http://dx.doi.org/10.1109/das49615.2020.9108942> (cit. on p. 2).
- [45] K. Liu. "Analysis of Features of Different Activation Functions." In: *2021 2nd International Conference on Computing and Data Science (CDS)*. IEEE, Jan. 2021. DOI: 10.1109/cds52072.2021.00078. URL: <http://dx.doi.org/10.1109/cds52072.2021.00078> (cit. on p. 2).

- [46] S. Serhat Kiliçarslan, K. Adem, and M. Çelik. “An overview of the activation functions used in deep learning algorithms.” In: *Journal of New Results in Science* 10.3 (Dec. 2021), pp. 75–88. ISSN: 1304-7981. DOI: 10.54187/jnrs.1011739. URL: <http://dx.doi.org/10.54187/jnrs.1011739> (cit. on p. 2).
- [47] G. K. Pandey and S. Srivastava. “ResNet-18 comparative analysis of various activation functions for image classification.” In: *2023 International Conference on Inventive Computation Technologies (ICICT)*. IEEE, Apr. 2023. DOI: 10.1109/iciict57646.2023.10134464. URL: <https://doi.org/10.1109/iciict57646.2023.10134464> (cit. on pp. 2, 6).
- [48] M. M. Noel, S. Bharadwaj, V. Muthiah-Nakarajan, P. Dutta, and G. B. Amali. *Biologically Inspired Oscillating Activation Functions Can Bridge the Performance Gap between Biological and Artificial Neurons*. 2021. DOI: 10.48550/ARXIV.2111.04020. URL: <https://arxiv.org/abs/2111.04020> (cit. on pp. 2, 3).
- [49] P. Ramachandran, B. Zoph, and Q. V. Le. *Searching for Activation Functions*. 2017. DOI: 10.48550/ARXIV.1710.05941. URL: <https://arxiv.org/abs/1710.05941> (cit. on pp. 2, 9, 55, 56, 85).
- [50] S. Eger, Y. Youssef, and I. Gurevych. “Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP tasks.” In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018. DOI: 10.18653/v1/d18-1472. URL: <https://doi.org/10.18653/v1/d18-1472> (cit. on pp. 2, 5).
- [51] L. Nanni, S. Brahnam, M. Paci, and S. Ghidoni. “Comparison of Different Convolutional Neural Network Activation Functions and Methods for Building Ensembles for Small to Midsize Medical Data Sets.” In: *Sensors* 22.16 (Aug. 2022), p. 6129. DOI: 10.3390/s22166129. URL: <https://doi.org/10.3390/s22166129> (cit. on p. 2).
- [52] K. Adem. “Impact of activation functions and number of layers on detection of exudates using circular Hough transform and convolutional neural networks.” In: *Expert Systems with Applications* 203 (Oct. 2022), p. 117583. DOI: 10.1016/j.eswa.2022.117583. URL: <https://doi.org/10.1016/j.eswa.2022.117583> (cit. on p. 2).
- [53] K. Adu, Y. Yu, J. Cai, I. Asare, and J. Quahin. “The influence of the activation function in a capsule network for brain tumor type classification.” In: *International Journal of Imaging Systems and Technology* 32.1 (Aug. 2021), pp. 123–143. DOI: 10.1002/ima.22638. URL: <https://doi.org/10.1002/ima.22638> (cit. on pp. 2, 60).
- [54] Z. A. Haq and Z. A. Jaffery. “Impact of activation functions and number of layers on the classification of fruits using CNN.” In: *Proceedings of the 2021 8th International Conference on Computing for Sustainable Global Development, INDIACom 2021*. IEEE, 2021, pp. 227–231. DOI: 10.1109/INDIACom51348.2021.00040. URL: <https://ieeexplore.ieee.org/document/9441246> (cit. on p. 2).
- [55] A. Nguyen, K. Pham, D. Ngo, T. Ngo, and L. Pham. “An Analysis of State-of-the-art Activation Functions For Supervised Deep Neural Network.” In: *2021 International Conference on System Science and Engineering (ICSSE)*. IEEE, Aug. 2021. DOI: 10.1109/icsse52999.2021.9538437. URL: <https://doi.org/10.1109/icsse52999.2021.9538437> (cit. on p. 2).
- [56] A. Mishra, P. Chandra, U. Ghose, and S. S. Sodhi. “Bi-modal derivative adaptive activation function sigmoidal feedforward artificial neural networks.” In: *Applied Soft Computing* 61 (Dec. 2017), pp. 983–994. DOI: 10.1016/j.asoc.2017.09.002. URL: <https://doi.org/10.1016/j.asoc.2017.09.002> (cit. on pp. 2, 53).
- [57] D. E. Ratnawati, Marjono, Widodo, and S. Anam. “Comparison of activation function on extreme learning machine (ELM) performance for classifying the active compound.” In: *SYMPOSIUM ON BIOMATHEMATICS 2019 (SYMOMATH 2019)*. AIP Publishing, 2020. DOI: 10.1063/5.0023872. URL: <https://doi.org/10.1063/5.0023872> (cit. on p. 2).
- [58] A. Dureja and P. Pawha. “Analysis of Nonlinear Activation Functions for Classification Tasks Using Convolutional Neural Networks.” In: *Lecture Notes in Electrical Engineering*. Springer Singapore, 2019, pp. 1179–1190. DOI: 10.1007/978-981-13-6772-4_103. URL: https://doi.org/10.1007/978-981-13-6772-4_103 (cit. on p. 2).
- [59] V. M. Vargas, D. Guijo-Rubio, P. A. Gutiérrez, and C. Hervás-Martínez. “ReLU-Based Activations: Analysis and Experimental Study for Deep Learning.” In: *Advances in Artificial Intelligence*. Springer International Publishing, 2021, pp. 33–43. DOI: 10.1007/978-3-030-85713-4_4. URL: https://doi.org/10.1007/978-3-030-85713-4_4 (cit. on p. 2).
- [60] Y. Singh, M. Saini, and Savita. “Impact and Performance Analysis of Various Activation Functions for Classification Problems.” In: *2023 IEEE International Conference on Contemporary Computing and Communications (InC4)*. IEEE, Apr. 2023. DOI: 10.1109/inc457730.2023.10263129. URL: <http://dx.doi.org/10.1109/InC457730.2023.10263129> (cit. on p. 2).
- [61] D. K.-H. Lai, E. S.-W. Cheng, B. P.-H. So, Y.-J. Mao, S. M.-Y. Cheung, D. S. K. Cheung, D. W.-C. Wong, and J. C.-W. Cheung. “Transformer Models and Convolutional Networks with Different Activation Functions for Swallow Classification Using Depth Video Data.” In: *Mathematics* 11.14 (July 2023), p. 3081. DOI: 10.3390/math11143081. URL: <https://doi.org/10.3390/math11143081> (cit. on p. 2).
- [62] E. Papavasileiou and B. Jansen. “The importance of the activation function in NeuroEvolution with FS-NEAT and FD-NEAT.” In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Nov. 2017. DOI: 10.1109/ssci.2017.8285328. URL: <https://doi.org/10.1109/ssci.2017.8285328> (cit. on p. 2).
- [63] W. H. Kang, J. Alam, and A. Fathan. “Investigation on activation functions for robust end-to-end spoofing attack detection system.” In: *2021 Edition of the Automatic Speaker Verification and Spoofing Countermeasures Challenge*. ISCA, Sept. 2021. DOI: 10.21437/asvspoof.2021-13. URL: <https://doi.org/10.21437/asvspoof.2021-13> (cit. on pp. 2, 44).
- [64] V. M. Vargas, P. A. Gutiérrez, J. Barbero-Gómez, and C. Hervás-Martínez. “Activation Functions for Convolutional Neural Networks: Proposals and Experimental Study.” In: *IEEE Transactions on Neural Networks and Learning Systems* 34.3 (Mar. 2023), pp. 1478–1488. DOI: 10.1109/tnnls.2021.3105444. URL: <https://doi.org/10.1109/tnnls.2021.3105444> (cit. on pp. 2, 65, 66).
- [65] R. H. K. Emanuel, P. D. Docherty, H. Lunt, and K. Möller. “The effect of activation functions on accuracy, convergence speed, and misclassification confidence in CNN text classification: a comprehensive exploration.” In: *The Journal of Supercomputing* 80.1 (June 2023), pp. 292–312. ISSN: 1573-0484. DOI: 10.1007/s11227-023-05441-7. URL: <http://dx.doi.org/10.1007/s11227-023-05441-7> (cit. on p. 2).
- [66] Z. Zhang, X. Li, Y. Yang, and Z. Shi. “Enhancing Deep Learning Models for Image Classification using Hybrid Activation Functions.” In: (Nov. 2023). DOI: 10.21203/rs.3.rs-3574353/v1. URL: <http://dx.doi.org/10.21203/rs.3.rs-3574353/v1> (cit. on p. 2).
- [67] B. Singh, S. Patel, A. Vijayvargiya, and R. Kumar. “Analyzing the impact of activation functions on the performance of the data-driven gait model.” In: *Results in Engineering* 18 (June 2023), p. 101029. ISSN: 2590-1230. DOI: 10.1016/j.rineng.2023.101029. URL: <http://dx.doi.org/10.1016/j.rineng.2023.101029> (cit. on pp. 2, 20).
- [68] K.-C. Lin, C.-H. Hu, and K.-C. Wang. “Innovative deep energy method for piezoelectricity problems.” In: *Applied Mathematical Modelling* 126 (Feb. 2024), pp. 405–419. ISSN: 0307-904X. DOI: 10.1016/j.apm.2023.11.006. URL: <http://dx.doi.org/10.1016/j.apm.2023.11.006> (cit. on p. 2).
- [69] N. T. Koh, A. Sharma, J. Xiao, X. Peng, and W. L. Woo. “Solar Irradiance Forecast using Long Short-Term Memory: A Comparative Analysis of Different Activation Functions.” In: *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Dec. 2022. DOI: 10.1109/ssci51031.2022.10022163. URL: <http://dx.doi.org/10.1109/ssci51031.2022.10022163> (cit. on p. 2).
- [70] S. H. Bhojani and N. Bhatt. “Performance Analysis of Activation Functions for Wheat Crop Yield Prediction.” In: *IOP Conference Series: Materials Science and Engineering* 1042.1 (Jan. 2021), p. 012015. ISSN: 1757-899X. DOI: 10.1088/1757-899x/1042/1/012015. URL: <http://dx.doi.org/10.1088/1757-899x/1042/1/012015> (cit. on p. 2).
- [71] L. A. Hurley, J. G. Restrepo, and S. E. Shaheen. *Tuning the activation function to optimize the forecast horizon of a reservoir computer*. 2023. DOI: 10.48550/ARXIV.2312.13151. URL: <https://arxiv.org/abs/2312.13151> (cit. on pp. 2, 32).
- [72] F. Makhrus. “The effect of amplitude modification in S-shaped activation functions on neural network regression.” In: *Neural Network World* 33.4 (2023), pp. 245–269. ISSN: 2336-4335. DOI: 10.14311/nnw.2023.33.015. URL: <http://dx.doi.org/10.14311/nnw.2023.33.015> (cit. on p. 2).

- [73] A. Mishra, P. Chandra, and U. Ghose. "A Non-monotonic Activation Function for Neural Networks Validated on Benchmark Tasks." In: *Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough*. Springer International Publishing, 2021, pp. 319–327. ISBN: 9783030682910. DOI: 10.1007/978-3-030-68291-0_25. URL: http://dx.doi.org/10.1007/978-3-030-68291-0_25 (cit. on pp. 2, 32).
- [74] M. H. Essai Ali, A. B. Abdel-Raman, and E. A. Badry. "Developing Novel Activation Functions Based Deep Learning LSTM for Classification." In: *IEEE Access* 10 (2022), pp. 97259–97275. ISSN: 2169-3536. DOI: 10.1109/access.2022.3205774. URL: <http://dx.doi.org/10.1109/ACCESS.2022.3205774> (cit. on pp. 2, 34).
- [75] D. J. Rumala, E. M. Yuniarno, R. F. Rachmadi, S. M. S. Nugroho, H. P. A. Tjahyaningtijas, Y. Adrianto, A. D. Sensusiaty, and I. K. E. Purnama. "Activation Functions Evaluation to Improve Performance of Convolutional Neural Network in Brain Disease Classification Based on Magnetic Resonance Images." In: *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*. IEEE, Nov. 2020. DOI: 10.1109/cenim51130.2020.9297862. URL: <http://dx.doi.org/10.1109/CENIM51130.2020.9297862> (cit. on p. 2).
- [76] L. Suciningtyas, R. Alfatikarani, M. B. F. Alan, F. M. Maimunir, and H. P. A. Tjahyaningtijas. "Activation Function Comparison On Potato Leaf Disease Classification Performance." In: *2023 Sixth International Conference on Vocational Education and Electrical Engineering (ICVEE)*. IEEE, Oct. 2023. DOI: 10.1109/icvee59738.2023.10348283. URL: <http://dx.doi.org/10.1109/ICVEE59738.2023.10348283> (cit. on p. 2).
- [77] H. Wang, L. Lu, S. S. null, and G. Huang. "Learning Specialized Activation Functions for Physics-Informed Neural Networks." In: *Communications in Computational Physics* 34.4 (June 2023), pp. 869–906. ISSN: 1991-7120. DOI: 10.4208/cicp.oa-2023-0058. URL: <http://dx.doi.org/10.4208/cicp.oa-2023-0058> (cit. on p. 2).
- [78] O. Pantalé. "Comparing Activation Functions in Machine Learning for Finite Element Simulations in Thermomechanical Forming." In: *Algorithms* 16.12 (Nov. 2023), p. 537. ISSN: 1999-4893. DOI: 10.3390/a16120537. URL: <http://dx.doi.org/10.3390/a16120537> (cit. on p. 2).
- [79] D. V. Dung, N. D. Song, P. S. Palar, and L. R. Zuhail. "On The Choice of Activation Functions in Physics-Informed Neural Network for Solving Incompressible Fluid Flows." In: *AIAA SCITECH 2023 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2023. DOI: 10.2514/6.2023-1803. URL: <http://dx.doi.org/10.2514/6.2023-1803> (cit. on p. 2).
- [80] X. Liu, J. Zhou, and H. Qian. "Comparison and Evaluation of Activation Functions in Term of Gradient Instability in Deep Neural Networks." In: *2019 Chinese Control And Decision Conference (CCDC)*. IEEE, June 2019. DOI: 10.1109/ccdc.2019.8832578. URL: <http://dx.doi.org/10.1109/CCDC.2019.8832578> (cit. on p. 2).
- [81] K. Ingole and N. Patil. "Performance Analysis of Various Activation Function on a Shallow Neural Network." In: *International Journal of Emerging Technologies and Innovative Research* 7.6 (June 2020), pp. 269–276. ISSN: 2349-5162. URL: <http://www.jetir.org/papers/JETIR2006041.pdf> (cit. on p. 2).
- [82] L. Nanni, A. Lumini, S. Ghidoni, and G. Maguolo. "Comparisons among different stochastic selections of activation layers for convolutional neural networks for health care." In: *Cognitive and Soft Computing Techniques for the Analysis of Healthcare Data*. Elsevier, 2022, pp. 151–164. DOI: 10.1016/b978-0-323-85751-2.00003-7. URL: <http://dx.doi.org/10.1016/b978-0-323-85751-2.00003-7> (cit. on p. 2).
- [83] R. Zhang, Y. Zhu, Z. Ge, H. Mu, D. Qi, and H. Ni. "Transfer Learning for Leaf Small Dataset Using Improved ResNet50 Network with Mixed Activation Functions." In: *Forests* 13.12 (Dec. 2022), p. 2072. ISSN: 1999-4907. DOI: 10.3390/f13122072. URL: <http://dx.doi.org/10.3390/f13122072> (cit. on p. 2).
- [84] A. Chaturvedi, N. Apoorva, M. S. Awasthi, S. Jyoti, D. P. Akarsha, S. Brunda, and C. S. Soumya. "Analyzing the Performance of Novel Activation Functions on Deep Learning Architectures." In: *Lecture Notes in Electrical Engineering*. Springer Nature Singapore, Dec. 2022, pp. 903–915. ISBN: 9789811954825. DOI: 10.1007/978-981-19-5482-5_76. URL: http://dx.doi.org/10.1007/978-981-19-5482-5_76 (cit. on p. 2).
- [85] D. Pedamonti. *Comparison of non-linear activation functions for deep neural networks on MNIST classification task*. 2018. DOI: 10.48550/ARXIV.1804.02763. URL: <https://arxiv.org/abs/1804.02763> (cit. on p. 2).
- [86] Y. Bai. "RELU-Function and Derived Function Review." In: *SHS Web of Conferences* 144 (2022). Ed. by A. Luqman, Q. Zhang, and W. Liu, p. 02006. ISSN: 2261-2424. DOI: 10.1051/shsconf/202214402006. URL: <http://dx.doi.org/10.1051/shsconf/202214402006> (cit. on p. 2).
- [87] F. Kamalov, A. Nazir, M. Safaraliev, A. K. Cherukuri, and R. Zgheib. "Comparative analysis of activation functions in neural networks." In: *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE, Nov. 2021. DOI: 10.1109/icecs53924.2021.9665646. URL: <http://dx.doi.org/10.1109/ICECS53924.2021.9665646> (cit. on p. 2).
- [88] M. M. Lau and K. Hann Lim. "Review of Adaptive Activation Function in Deep Neural Network." In: *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*. IEEE, Dec. 2018. DOI: 10.1109/iecbes.2018.8626714. URL: <http://dx.doi.org/10.1109/IECBES.2018.8626714> (cit. on p. 2).
- [89] A. Dubowski. *Activation function impact on Sparse Neural Networks*. 2020. DOI: 10.48550/ARXIV.2010.05943. URL: <https://arxiv.org/abs/2010.05943> (cit. on p. 2).
- [90] I. A. Kandhro, M. Uddin, S. Hussain, T. J. Chaudhery, M. Shorufuzzaman, H. Meshref, M. Albalhaq, R. Alsaqour, and O. I. Khalaf. "Impact of Activation, Optimization, and Regularization Methods on the Facial Expression Model Using CNN." In: *Computational Intelligence and Neuroscience* 2022 (June 2022). Ed. by M. Z. Asghar, pp. 1–9. ISSN: 1687-5265. DOI: 10.1155/2022/3098604. URL: <http://dx.doi.org/10.1155/2022/3098604> (cit. on p. 2).
- [91] E. C. Seyrek and M. Uysal. "A comparative analysis of various activation functions and optimizers in a convolutional neural network for hyperspectral image classification." In: *Multimedia Tools and Applications* (Nov. 2023). ISSN: 1573-7721. DOI: 10.1007/s11042-023-17546-5. URL: <http://dx.doi.org/10.1007/s11042-023-17546-5> (cit. on p. 2).
- [92] C. Bircanoglu and N. Arica. "A comparison of activation functions in artificial neural networks." In: *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, May 2018. DOI: 10.1109/siu.2018.8404724. URL: <http://dx.doi.org/10.1109/SIU.2018.8404724> (cit. on p. 2).
- [93] P. Tatraiya, H. Priyadarshi, K. Singh, D. Mishra, and A. Shrivastava. "Applicative Analysis Of Activation Functions For Pneumonia Detection Using Convolutional Neural Networks." In: *2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*. IEEE, May 2023. DOI: 10.1109/globconet56651.2023.10149937. URL: <http://dx.doi.org/10.1109/GlobConET56651.2023.10149937> (cit. on p. 2).
- [94] W. Qiu, C. He, G. Zheng, Q. Yi, and G. Chen. "Activation Function Dependence of Data-Driven Spectra Prediction of Nanostructures." In: *Advanced Theory and Simulations* 6.5 (Feb. 2023). ISSN: 2513-0390. DOI: 10.1002/adts.202200867. URL: <http://dx.doi.org/10.1002/adts.202200867> (cit. on p. 2).
- [95] V. K. Kayala and P. Kodali. "Performance Analysis of Activation Functions on Convolutional Neural Networks Using Cloud GPU." In: *Advances in Communications, Signal Processing, and VLSI*. Springer Singapore, 2021, pp. 35–48. ISBN: 9789813340589. DOI: 10.1007/978-981-33-4058-9_4. URL: http://dx.doi.org/10.1007/978-981-33-4058-9_4 (cit. on p. 2).
- [96] L. Xu and C. L. Philip Chen. "Comparison and Combination of Activation Functions in Broad Learning System." In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Oct. 2020. DOI: 10.1109/smc42975.2020.9282871. URL: <http://dx.doi.org/10.1109/SMC42975.2020.9282871> (cit. on pp. 2, 11, 17, 26, 35, 36).
- [97] O. H. Purba, E. A. Sarwoko, Khadjjah, Suhartono, and A. Wibowo. "Classification of liver cancer with microrna data using the deep neural network (DNN) method." In: *Journal of Physics: Conference Series* 1524.1 (Apr. 2020), p. 012129. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1524/1/012129. URL: <http://dx.doi.org/10.1088/1742-6596/1524/1/012129> (cit. on p. 2).
- [98] A. Salam, A. E. Hibaoui, and A. Saif. "A comparison of activation functions in multilayer neural network for predicting the production and consumption of electricity power." In: *International Journal of Electrical and Computer Engineering (IJECE)* 11.1 (Feb. 2021), p. 163. ISSN: 2088-8708. DOI: 10.11591/ijece.v11i1.pp163-170. URL: <http://dx.doi.org/10.11591/ijece.v11i1.pp163-170> (cit. on p. 2).

- [99] A. S. Lutakamale and Y. Z. Manyesela. “The Influence of Non-learnable Activation Functions on the Positioning Performance of Deep Learning-Based Fingerprinting Models Trained with Small CSI Sample Sizes.” In: *Transactions of the Indian National Academy of Engineering* 7.3 (July 2022), pp. 1059–1067. ISSN: 2662-5423. DOI: 10.1007/s41403-022-00347-x. URL: <http://dx.doi.org/10.1007/s41403-022-00347-x> (cit. on p. 2).
- [100] A. Nader and D. Azar. “Evolution of Activation Functions: An Empirical Investigation.” In: *ACM Transactions on Evolutionary Learning and Optimization* 1.2 (June 2021), pp. 1–36. DOI: 10.1145/3464384. URL: <https://doi.org/10.1145/3464384> (cit. on p. 2).
- [101] M. Basirat and P. M. Roth. *The Quest for the Golden Activation Function*. 2018. DOI: 10.48550/ARXIV.1808.00783. URL: <https://arxiv.org/abs/1808.00783> (cit. on pp. 2, 18, 29).
- [102] Basirat, Mina, Jammer, Alexandra, and Roth, Peter M. “The Quest for the Golden Activation Function.” In: *Proceedings of ARW & OAGM Workshop 2019*. Verlag der Technischen Universität Graz, 2019. DOI: 10.3217/978-3-85125-663-5-41. URL: <https://openlib.tugraz.at/download.php?id=5d09dba127371&location=medra> (cit. on p. 2).
- [103] H. A. Mayer and R. Schwaiger. “Differentiation of neuron types by evolving activation function templates for artificial neural networks.” In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN’02 (Cat. No.02CH37290)*. IEEE, 2002. DOI: 10.1109/ijcnn.2002.1007787. URL: <https://doi.org/10.1109/ijcnn.2002.1007787> (cit. on pp. 2, 77).
- [104] A. Hagg, M. Mensing, and A. Asteroth. “Evolving parsimonious networks by mixing activation functions.” In: *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, July 2017. DOI: 10.1145/3071178.3071275. URL: <https://doi.org/10.1145/3071178.3071275> (cit. on p. 2).
- [105] K. Knezevic, J. Fulir, D. Jakobovic, S. Picsek, and M. Durasevic. “NeuroSCA: Evolving Activation Functions for Side-Channel Analysis.” In: *IEEE Access* 11 (2023), pp. 284–299. DOI: 10.1109/access.2022.3232064. URL: <https://doi.org/10.1109/access.2022.3232064> (cit. on p. 2).
- [106] Y. Liu and X. Yao. “Evolutionary design of artificial neural networks with different nodes.” In: *Proceedings of IEEE International Conference on Evolutionary Computation*. IEEE, 1996. DOI: 10.1109/icec.1996.542681. URL: <https://doi.org/10.1109/icec.1996.542681> (cit. on p. 2).
- [107] P. Cui and K. C. Wiese. “EvoDNN - Evolving Weights, Biases, and Activation Functions in a Deep Neural Network.” In: *2022 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE, Aug. 2022. DOI: 10.1109/cibcb55180.2022.9863054. URL: <https://doi.org/10.1109/cibcb55180.2022.9863054> (cit. on p. 2).
- [108] P. Cui, B. Shabash, and K. C. Wiese. “EvoDNN - An Evolutionary Deep Neural Network with Heterogeneous Activation Functions.” In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, June 2019. DOI: 10.1109/cec.2019.8789964. URL: <https://doi.org/10.1109/cec.2019.8789964> (cit. on p. 2).
- [109] K. Vijayaprabakaran and K. Sathiyamurthy. “Towards activation function search for long short-term model network: A differential evolution based approach.” In: *Journal of King Saud University - Computer and Information Sciences* 34.6 (June 2022), pp. 2637–2650. DOI: 10.1016/j.jksuci.2020.04.015. URL: <https://doi.org/10.1016/j.jksuci.2020.04.015> (cit. on pp. 2, 35).
- [110] D. O’Neill, B. Xue, and M. Zhang. “Co-evolution of Novel Tree-Like ANNs and Activation Functions: An Observational Study.” In: *AI 2018: Advances in Artificial Intelligence*. Springer International Publishing, 2018, pp. 616–629. DOI: 10.1007/978-3-030-03991-2_56. URL: https://doi.org/10.1007/978-3-030-03991-2_56 (cit. on p. 2).
- [111] M. Sipper. “Neural Networks with À La Carte Selection of Activation Functions.” In: *SN Computer Science* 2.6 (Sept. 2021). ISSN: 2661-8907. DOI: 10.1007/s42979-021-00885-1. URL: <http://dx.doi.org/10.1007/s42979-021-00885-1> (cit. on pp. 2, 20).
- [112] M. Salimi, M. Loni, and M. Sirjani. “Learning Activation Functions for Adversarial Attack Resilience in CNNs.” In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2023, pp. 203–214. ISBN: 9783031425059. DOI: 10.1007/978-3-031-42505-9_18. URL: http://dx.doi.org/10.1007/978-3-031-42505-9_18 (cit. on p. 2).
- [113] M. Salimi, M. Loni, M. Sirjani, A. Cicchetti, and S. Abbaspour Asadollah. “SARAF: Searching for Adversarial Robust Activation Functions.” In: *Proceedings of the 2023 6th International Conference on Machine Vision and Applications*. ICMVA 2023. ACM, Mar. 2023. DOI: 10.1145/3589572.3589598. URL: <http://dx.doi.org/10.1145/3589572.3589598> (cit. on p. 2).
- [114] Y. Li, T. Geng, S. Stein, A. Li, and H. Yu. “GAFF: Searching Activation Functions for Binary Neural Networks Through Genetic Algorithm.” In: *Tsinghua Science and Technology* 28.1 (Feb. 2023), pp. 207–220. ISSN: 1007-0214. DOI: 10.26599/tst.2021.9010084. URL: <http://dx.doi.org/10.26599/TST.2021.9010084> (cit. on p. 2).
- [115] J. Chen. *Combinatorially Generated Piecewise Activation Functions*. 2016. DOI: 10.48550/ARXIV.1605.05216. URL: <https://arxiv.org/abs/1605.05216> (cit. on p. 2).
- [116] K. Vijayaprabakaran and K. Sathiyamurthy. “Neuroevolution based hierarchical activation function for long short-term model network.” In: *Journal of Ambient Intelligence and Humanized Computing* 12.12 (Jan. 2021), pp. 10757–10768. ISSN: 1868-5145. DOI: 10.1007/s12652-020-02889-w. URL: <http://dx.doi.org/10.1007/s12652-020-02889-w> (cit. on pp. 2, 37).
- [117] Y. Pan, Y. Wang, P. Zhou, Y. Yan, and D. Guo. “Activation functions selection for BP neural network model of ground surface roughness.” In: *Journal of Intelligent Manufacturing* 31.8 (Jan. 2020), pp. 1825–1836. DOI: 10.1007/s10845-020-01538-5. URL: <https://doi.org/10.1007/s10845-020-01538-5> (cit. on pp. 2, 35, 37).
- [118] A. Marchisio, M. A. Hanif, S. Rehman, M. Martina, and M. Shafique. *A Methodology for Automatic Selection of Activation Functions to Design Hybrid Deep Neural Networks*. 2018. DOI: 10.48550/ARXIV.1811.03980. URL: <https://arxiv.org/abs/1811.03980> (cit. on p. 2).
- [119] R. P. Tripathi, M. Tiwari, A. Dhawan, A. Sharma, and S. K. Jha. “A Survey on Efficient Realization of Activation Functions of Artificial Neural Network.” In: *2021 Asian Conference on Innovation in Technology (ASIANCON)*. IEEE, Aug. 2021. DOI: 10.1109/asiancon51346.2021.9544754. URL: <https://doi.org/10.1109/asiancon51346.2021.9544754> (cit. on p. 2).
- [120] S. Bouguezzi, H. Faiedh, and C. Souani. “Hardware Implementation of Tanh Exponential Activation Function using FPGA.” In: *2021 18th International Multi-Conference on Systems, Signals & Devices (SSD)*. IEEE, Mar. 2021. DOI: 10.1109/ssd52085.2021.9429506. URL: <https://doi.org/10.1109/ssd52085.2021.9429506> (cit. on p. 2).
- [121] L. Li, S. Zhang, and J. Wu. “An Efficient Hardware Architecture for Activation Function in Deep Learning Processor.” In: *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*. IEEE, June 2018. DOI: 10.1109/icivc.2018.8492754. URL: <https://doi.org/10.1109/icivc.2018.8492754> (cit. on p. 2).
- [122] C.-H. Tsai, Y.-T. Chih, W. H. Wong, and C.-Y. Lee. “A Hardware-Efficient Sigmoid Function With Adjustable Precision for a Neural Network System.” In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 62.11 (Nov. 2015), pp. 1073–1077. DOI: 10.1109/tcsii.2015.2456531. URL: <https://doi.org/10.1109/tcsii.2015.2456531> (cit. on p. 2).
- [123] A. H. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi. “Efficient hardware implementation of the hyperbolic tangent sigmoid function.” In: *2009 IEEE International Symposium on Circuits and Systems*. IEEE, May 2009. DOI: 10.1109/iscas.2009.5118213. URL: <https://doi.org/10.1109/iscas.2009.5118213> (cit. on p. 2).
- [124] R. Pogiri, S. Ari, and K. K. Mahapatra. “Design and FPGA Implementation of the LUT based Sigmoid Function for DNN Applications.” In: *2022 IEEE International Symposium on Smart Electronic Systems (ISES)*. IEEE, Dec. 2022. DOI: 10.1109/ises54909.2022.00090. URL: <https://doi.org/10.1109/ises54909.2022.00090> (cit. on p. 2).
- [125] F. M. Shakiba and M. Zhou. “Novel Analog Implementation of a Hyperbolic Tangent Neuron in Artificial Neural Networks.” In: *IEEE Transactions on Industrial Electronics* 68.11 (Nov. 2021), pp. 10856–10867. DOI: 10.1109/tie.2020.3034856. URL: <https://doi.org/10.1109/tie.2020.3034856> (cit. on p. 2).

- [126] Y. Xie, A. N. J. Raj, Z. Hu, S. Huang, Z. Fan, and M. Joler. “A Twofold Lookup Table Architecture for Efficient Approximation of Activation Functions.” In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28.12 (Dec. 2020), pp. 2540–2550. DOI: 10.1109/tvlsi.2020.3015391. URL: <https://doi.org/10.1109/tvlsi.2020.3015391> (cit. on p. 2).
- [127] P. Priyanka, G. K. Nisarga, and S. Raghuram. “CMOS Implementations of Rectified Linear Activation Function.” In: *VLSI Design and Test*. Springer Singapore, 2019, pp. 121–129. ISBN: 9789811359507. DOI: 10.1007/978-981-13-5950-7_11. URL: http://dx.doi.org/10.1007/978-981-13-5950-7_11 (cit. on p. 2).
- [128] S.-Y. Lin and J.-C. Chiang. “Low-area architecture design of multi-mode activation functions with controllable maximum absolute error for neural network applications.” In: *Microprocessors and Microsystems* 103 (Nov. 2023), p. 104952. ISSN: 0141-9331. DOI: 10.1016/j.micpro.2023.104952. URL: <http://dx.doi.org/10.1016/j.micpro.2023.104952> (cit. on pp. 2, 57).
- [129] V. Shatravin, D. Shashev, and S. Shidlovskiy. “Sigmoid Activation Implementation for Neural Networks Hardware Accelerators Based on Reconfigurable Computing Environments for Low-Power Intelligent Systems.” In: *Applied Sciences* 12.10 (May 2022), p. 5216. ISSN: 2076-3417. DOI: 10.3390/app12105216. URL: <http://dx.doi.org/10.3390/app12105216> (cit. on p. 2).
- [130] L. Derczynski. *Power Consumption Variation over Activation Functions*. 2020. DOI: 10.48550/ARXIV.2006.07237. URL: <https://arxiv.org/abs/2006.07237> (cit. on p. 2).
- [131] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. “Efficient Neural Network Robustness Certification with General Activation Functions.” In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/d04863f100d59b3eb688a11f95b0ae60-Paper.pdf (cit. on p. 3).
- [132] A. Dinu and M. Cirstea. “A Digital Neural Network FPGA Direct Hardware Implementation Algorithm.” In: *2007 IEEE International Symposium on Industrial Electronics*. IEEE, June 2007. DOI: 10.1109/isie.2007.4374572. URL: <https://doi.org/10.1109/isie.2007.4374572> (cit. on p. 3).
- [133] A. Dinu, M. N. Cirstea, and S. E. Cirstea. “Direct Neural-Network Hardware-Implementation Algorithm.” In: *IEEE Transactions on Industrial Electronics* 57.5 (May 2010), pp. 1845–1848. DOI: 10.1109/tie.2009.2033097. URL: <https://doi.org/10.1109/tie.2009.2033097> (cit. on p. 3).
- [134] J. Chen and Z. Pan. *Saturated Non-Monotonic Activation Functions*. 2023. DOI: 10.48550/ARXIV.2305.07537. URL: <https://arxiv.org/abs/2305.07537> (cit. on p. 3).
- [135] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519. URL: <https://doi.org/10.1037/h0042519> (cit. on p. 3).
- [136] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016 (cit. on pp. 3, 15, 39).
- [137] G. S. da S. Gomes, T. B. Ludermir, and L. M. M. R. Lima. “Comparison of new activation functions in neural network for forecasting financial time series.” In: *Neural Computing and Applications* 20.3 (June 2010), pp. 417–439. ISSN: 1433-3058. DOI: 10.1007/s00521-010-0407-3. URL: <http://dx.doi.org/10.1007/s00521-010-0407-3> (cit. on pp. 3, 7, 8).
- [138] T. E. Simos and C. Tsitouras. “Efficiently inaccurate approximation of hyperbolic tangent used as transfer function in artificial neural networks.” In: *Neural Computing and Applications* 33.16 (Mar. 2021), pp. 10227–10233. DOI: 10.1007/s00521-021-05787-0. URL: <https://doi.org/10.1007/s00521-021-05787-0> (cit. on p. 3).
- [139] G. Dudek. “Data-Driven Learning of Feedforward Neural Networks with Different Activation Functions.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2021, pp. 66–77. ISBN: 9783030879860. DOI: 10.1007/978-3-030-87986-0_6. URL: http://dx.doi.org/10.1007/978-3-030-87986-0_6 (cit. on pp. 3, 35).
- [140] D. W. Edwards and I. Dinc. “LRTanH: Substitution for the Activation Function Derivative during Back Propagation.” In: *2019 SoutheastCon*. IEEE, Apr. 2019. DOI: 10.1109/southeastcon42311.2019.9020655. URL: <http://dx.doi.org/10.1109/SoutheastCon42311.2019.9020655> (cit. on p. 3).
- [141] K. Sunat, C. Lursinsap, and C.-H. H. Chu. “The p-recursive piecewise polynomial sigmoid generators and first-order algorithms for multilayer tanh-like neurons.” In: *Neural Computing and Applications* 16.1 (Apr. 2006), pp. 33–47. ISSN: 1433-3058. DOI: 10.1007/s00521-006-0046-x. URL: <http://dx.doi.org/10.1007/s00521-006-0046-x> (cit. on p. 3).
- [142] H. Kwan. “Simple sigmoid-like activation function suitable for digital hardware implementation.” In: *Electronics Letters* 28.15 (1992), p. 1379. ISSN: 0013-5194. DOI: 10.1049/el:19920877. URL: <http://dx.doi.org/10.1049/el:19920877> (cit. on p. 3).
- [143] M. Zhang, S. Vassiliadis, and J. Delgado-Frias. “Sigmoid generators for neural computing using piecewise approximations.” In: *IEEE Transactions on Computers* 45.9 (1996), pp. 1045–1049. ISSN: 0018-9340. DOI: 10.1109/12.537127. URL: <http://dx.doi.org/10.1109/12.537127> (cit. on p. 3).
- [144] B. Xu, R. Huang, and M. Li. *Revise Saturated Activation Functions*. 2016. DOI: 10.48550/ARXIV.1602.05980. URL: <https://arxiv.org/abs/1602.05980> (cit. on pp. 3, 5).
- [145] X. Glorot and Y. Bengio. “Understanding the difficulty of training deep feedforward neural networks.” In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Y. W. Teh and M. Titterton. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256. URL: <http://proceedings.mlr.press/v9/glorot10a.html> (cit. on p. 3).
- [146] D. B. Mulindwa and S. Du. “An n-Sigmoid Activation Function to Improve the Squeeze-and-Excitation for 2D and 3D Deep Networks.” In: *Electronics* 12.4 (Feb. 2023), p. 911. ISSN: 2079-9292. DOI: 10.3390/electronics12040911. URL: <http://dx.doi.org/10.3390/electronics12040911> (cit. on p. 4).
- [147] H. Arai and H. Imamura. “Spin-wave coupled spin torque oscillators for artificial neural network.” In: *Journal of Applied Physics* 124.15 (Oct. 2018). ISSN: 1089-7550. DOI: 10.1063/1.5040020. URL: <http://dx.doi.org/10.1063/1.5040020> (cit. on p. 4).
- [148] J. Han and C. Moraga. “The influence of the sigmoid function parameters on the speed of backpropagation learning.” In: *From Natural to Artificial Neural Computation*. Springer Berlin Heidelberg, 1995, pp. 195–201. ISBN: 9783540492887. DOI: 10.1007/3-540-59497-3_175. URL: http://dx.doi.org/10.1007/3-540-59497-3_175 (cit. on p. 4).
- [149] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition.” In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791. URL: <https://doi.org/10.1109/5.726791> (cit. on pp. 4, 54).
- [150] S. S. Sodhi and P. Chandra. “Bi-modal derivative activation function for sigmoidal feedforward networks.” In: *Neurocomputing* 143 (Nov. 2014), pp. 182–196. DOI: 10.1016/j.neucom.2014.06.007. URL: <https://doi.org/10.1016/j.neucom.2014.06.007> (cit. on p. 4).
- [151] T. T. Sivri, N. P. Akman, and A. Berkol. “Multiclass Classification Using Arctangent Activation Function and Its Variations.” In: *2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. IEEE, June 2022. DOI: 10.1109/ecai54874.2022.9847486. URL: <https://doi.org/10.1109/ecai54874.2022.9847486> (cit. on p. 4).
- [152] J. Kamruzzaman and S. Aziz. “A note on activation function in multilayer feedforward learning.” In: *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290)*. IJCNN-02. IEEE, 2002. DOI: 10.1109/ijcnn.2002.1005526. URL: <http://dx.doi.org/10.1109/IJCNN.2002.1005526> (cit. on pp. 4, 32).
- [153] T. Tümer-Sivri, N. Pervan-Akman, and A. Berkol. “The Impact of Irrationals on the Range of Arctan Activation Function for Deep Learning Models.” In: (May 2023). DOI: 10.20944/preprints202305.1245.v1. URL: <http://dx.doi.org/10.20944/preprints202305.1245.v1> (cit. on pp. 4, 11).
- [154] Y. Koçak and G. Üstündağ Şiray. “New activation functions for single layer feedforward neural network.” In: *Expert Systems with Applications* 164 (Feb. 2021), p. 113977. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2020.113977. URL: <http://dx.doi.org/10.1016/j.eswa.2020.113977> (cit. on pp. 4–6, 10, 25).

- [155] Y. Qin, X. Wang, and J. Zou. "The Optimized Deep Belief Networks With Improved Logistic Sigmoid Units and Their Application in Fault Diagnosis for Planetary Gearboxes of Wind Turbines." In: *IEEE Transactions on Industrial Electronics* 66.5 (May 2019), pp. 3814–3824. DOI: 10.1109/tie.2018.2856205. URL: <https://doi.org/10.1109/tie.2018.2856205> (cit. on p. 5).
- [156] M. Roodschild, J. Gotay Sardiñas, and A. Will. "A new approach for the vanishing gradient problem on sigmoid activation." In: *Progress in Artificial Intelligence* 9.4 (Oct. 2020), pp. 351–360. ISSN: 2192-6360. DOI: 10.1007/s13748-020-00218-y. URL: <http://dx.doi.org/10.1007/s13748-020-00218-y> (cit. on p. 5).
- [157] D. Li and Y. Zhou. "Soft-Root-Sign: A New Bounded Neural Activation Function." In: *Pattern Recognition and Computer Vision*. Springer International Publishing, 2020, pp. 310–319. DOI: 10.1007/978-3-030-60636-7_26. URL: https://doi.org/10.1007/978-3-030-60636-7_26 (cit. on p. 5).
- [158] A. Krizhevsky. "Learning multiple layers of features from tiny images." MA thesis. Department of Computer Science, University of Toronto, 2009 (cit. on pp. 5, 9, 10, 13, 14, 24, 27, 29, 32, 49, 50, 53, 56, 57, 63, 68, 75).
- [159] I. Ohn and Y. Kim. "Smooth Function Approximation by Deep Neural Networks with General Activation Functions." In: *Entropy* 21.7 (June 2019), p. 627. ISSN: 1099-4300. DOI: 10.3390/e21070627. URL: <http://dx.doi.org/10.3390/e21070627> (cit. on p. 6).
- [160] M. Klimek and M. Perelstein. "Neural network-based approach to phase space integration." In: *SciPost Physics* 9.4 (Oct. 2020). ISSN: 2542-4653. DOI: 10.21468/scipostphys.9.4.053. URL: <http://dx.doi.org/10.21468/SciPostPhys.9.4.053> (cit. on p. 6).
- [161] S. Kong and M. Takatsuka. "Hexpo: A vanishing-proof activation function." In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, May 2017. DOI: 10.1109/ijcnn.2017.7966168. URL: <https://doi.org/10.1109/ijcnn.2017.7966168> (cit. on p. 6).
- [162] H. Hazimeh, N. Ponomareva, P. Mol, Z. Tan, and R. Mazumder. "The tree ensemble layer: differentiability meets conditional computation." In: *Proceedings of the 37th International Conference on Machine Learning, ICML'20*. JMLR.org, 2020. URL: <http://proceedings.mlr.press/v119/hazimeh20a/hazimeh20a.pdf> (cit. on p. 6).
- [163] D. L. Elliott. *A better Activation Function for Artificial Neural Networks*. Tech. rep. 1993. URL: <https://www.researchgate.net/publication/277299531> (cit. on p. 6).
- [164] H. Burhani, W. Feng, and G. Hu. "Denoising AutoEncoder in Neural Networks with Modified Elliott Activation Function and Sparsity-Favoring Cost Function." In: *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence*. IEEE, July 2015. DOI: 10.1109/acit-csi.2015.67. URL: <http://dx.doi.org/10.1109/ACIT-CSI.2015.67> (cit. on pp. 6, 32).
- [165] M. Kaytan, İ. B. Aydılek, C. Yeroğlu, and A. Karci. "Sigmoid-Gumbel: Yeni Bir Hibrit Aktivasyon Fonksiyonu." In: *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi* 11.1 (Mar. 2022), pp. 29–45. ISSN: 2147-3129. DOI: 10.17798/bitlisfen.990508. URL: <http://dx.doi.org/10.17798/bitlisfen.990508> (cit. on p. 7).
- [166] A. Kumar and S. S. Sodhi. "Image Segmentation on Convolutional Neural Network (CNN) using Some New Activation Functions." In: *Communications in Mathematics and Applications* 14.2 (Sept. 2023), pp. 949–968. ISSN: 0975-8607. DOI: 10.26713/cma.v14i2.2152. URL: <http://dx.doi.org/10.26713/cma.v14i2.2152> (cit. on p. 7).
- [167] C. Közkurt, S. Kiliçarslan, S. Baş, and A. Elen. " α SechSig and α TanhSig: two novel non-monotonic activation functions." In: *Soft Computing* 27.24 (Oct. 2023), pp. 18451–18467. ISSN: 1433-7479. DOI: 10.1007/s00500-023-09279-2. URL: <http://dx.doi.org/10.1007/s00500-023-09279-2> (cit. on p. 7).
- [168] C. Cai, Y. Xu, D. Ke, and K. Su. "Deep Neural Networks with Multistate Activation Functions." In: *Computational Intelligence and Neuroscience* 2015 (2015), pp. 1–10. ISSN: 1687-5273. DOI: 10.1155/2015/721367. URL: <http://dx.doi.org/10.1155/2015/721367> (cit. on p. 8).
- [169] W. Duch and N. Jankowski. *Survey of Neural Transfer Functions*. 1999 (cit. on p. 8).
- [170] M. C. N. Guevara, V. G. S. Cruz, O. O. V. Vergara, M. Nandayapa, H. d. J. D. Ochoa, and H. J. A. Sossa. "Study of the Effect of Combining Activation Functions in a Convolutional Neural Network." In: *IEEE Latin America Transactions* 19.5 (May 2021), pp. 844–852. ISSN: 1548-0992. DOI: 10.1109/tla.2021.9448319. URL: <http://dx.doi.org/10.1109/TLA.2021.9448319> (cit. on pp. 8, 26, 46).
- [171] S. Elfving, E. Uchibe, and K. Doya. "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning." In: *Neural Networks* 107 (Nov. 2018), pp. 3–11. DOI: 10.1016/j.neunet.2017.12.012. URL: <https://doi.org/10.1016/j.neunet.2017.12.012> (cit. on pp. 9, 12, 55).
- [172] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A large-scale hierarchical image database." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2009. DOI: 10.1109/cvpr.2009.5206848. URL: <https://doi.org/10.1109/cvpr.2009.5206848> (cit. on pp. 9, 13, 63).
- [173] O. Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge." In: *International Journal of Computer Vision* 115.3 (Apr. 2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y. URL: <https://doi.org/10.1007/s11263-015-0816-y> (cit. on pp. 9, 13, 53, 63).
- [174] M. Tanaka. "Weighted sigmoid gate unit for an activation function of deep neural network." In: *Pattern Recognition Letters* 135 (July 2020), pp. 354–359. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2020.05.017. URL: <http://dx.doi.org/10.1016/j.patrec.2020.05.017> (cit. on p. 9).
- [175] D. Hendrycks and K. Gimpel. *Gaussian Error Linear Units (GELUs)*. 2016. DOI: 10.48550/ARXIV.1606.08415. URL: <https://arxiv.org/abs/1606.08415> (cit. on p. 9).
- [176] M. Lee. *GELU Activation Function in Deep Learning: A Comprehensive Mathematical Analysis and Performance*. 2023. DOI: 10.48550/ARXIV.2305.12073. URL: <https://arxiv.org/abs/2305.12073> (cit. on p. 9).
- [177] J. Kang, R. Liu, Y. Li, Q. Liu, P. Wang, Q. Zhang, and D. Zhou. "An Improved 3D Human Pose Estimation Model Based on Temporal Convolution with Gaussian Error Linear Units." In: *2022 8th International Conference on Virtual Reality (ICVR)*. IEEE, May 2022. DOI: 10.1109/icvr55215.2022.9848068. URL: <https://doi.org/10.1109/icvr55215.2022.9848068> (cit. on pp. 9, 44).
- [178] C. Yu and Z. Su. *Symmetrical Gaussian Error Linear Units (SGELUs)*. 2019. DOI: 10.48550/ARXIV.1911.03925. URL: <https://arxiv.org/abs/1911.03925> (cit. on p. 9).
- [179] Z. Wu, H. Yu, L. Zhang, and Y. Sui. "The Adaptive Quadratic Linear Unit (AQuLU): Adaptive Non Monotonic Piecewise Activation Function." In: *Tehnicki vjesnik - Technical Gazette* 30.5 (Oct. 2023). ISSN: 1848-6339. DOI: 10.17559/tv-20230614000735. URL: <http://dx.doi.org/10.17559/tv-20230614000735> (cit. on pp. 9–11, 57).
- [180] A. Vagerwal. *Deeper Learning with CoLU Activation*. 2021. DOI: 10.48550/ARXIV.2112.12078. URL: <https://arxiv.org/abs/2112.12078> (cit. on p. 10).
- [181] M. Kaytan, İ. B. Aydılek, and C. Yeroğlu. "Gish: a novel activation function for image classification." In: *Neural Computing and Applications* 35.34 (Sept. 2023), pp. 24259–24281. ISSN: 1433-3058. DOI: 10.1007/s00521-023-09035-5. URL: <http://dx.doi.org/10.1007/s00521-023-09035-5> (cit. on p. 10).
- [182] L. Deng. "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]." In: *IEEE Signal Processing Magazine* 29.6 (Nov. 2012), pp. 141–142. DOI: 10.1109/msp.2012.2211477. URL: <https://doi.org/10.1109/msp.2012.2211477> (cit. on pp. 10, 13, 29, 30, 49, 56, 63, 68, 70, 75).
- [183] H. Zhu, H. Zeng, J. Liu, and X. Zhang. "Logish: A new nonlinear nonmonotonic activation function for convolutional neural network." In: *Neurocomputing* 458 (Oct. 2021), pp. 490–499. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.06.067. URL: <http://dx.doi.org/10.1016/j.neucom.2021.06.067> (cit. on pp. 10, 11).
- [184] P. Naveen. "Phish: A Novel Hyper-Optimizable Activation Function." In: (Jan. 2022). DOI: 10.36227/techrxiv.17283824.v2. URL: <http://dx.doi.org/10.36227/techrxiv.17283824.v2> (cit. on p. 11).

- [185] S. Jianlin. *A brief discussion on the design of activation functions in neural networks*. 2017. URL: <https://spaces.ac.cn/archives/4647> (visited on 01/28/2024) (cit. on p. 11).
- [186] M. A. Mercioni, A. M. Tat, and S. Holban. “Improving the Accuracy of Deep Neural Networks Through Developing New Activation Functions.” In: *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, Sept. 2020. DOI: 10.1109/iccp51029.2020.9266162. URL: <http://dx.doi.org/10.1109/ICCP51029.2020.9266162> (cit. on pp. 11, 57).
- [187] K. Wong, R. Dornberger, and T. Hanne. “An analysis of weight initialization methods in connection with different activation functions for feedforward neural networks.” In: *Evolutionary Intelligence* (Nov. 2022). DOI: 10.1007/s12065-022-00795-y. URL: <https://doi.org/10.1007/s12065-022-00795-y> (cit. on p. 11).
- [188] S. Verma, A. Chug, and A. P. Singh. “Revisiting activation functions: empirical evaluation for image understanding and classification.” In: *Multimedia Tools and Applications* (July 2023). ISSN: 1573-7721. DOI: 10.1007/s11042-023-16159-2. URL: <http://dx.doi.org/10.1007/s11042-023-16159-2> (cit. on p. 12).
- [189] S. Hayou, A. Doucet, and J. Rousseau. *On the Selection of Initialization and Activation Function for Deep Neural Networks*. 2018. DOI: 10.48550/ARXIV.1805.08266. URL: <https://arxiv.org/abs/1805.08266> (cit. on p. 12).
- [190] A. N. S. Njikam and H. Zhao. “A novel activation function for multilayer feed-forward neural networks.” In: *Applied Intelligence* 45.1 (Jan. 2016), pp. 75–82. DOI: 10.1007/s10489-015-0744-0. URL: <https://doi.org/10.1007/s10489-015-0744-0> (cit. on p. 12).
- [191] S. K. Roy, S. Manna, S. R. Dubey, and B. B. Chaudhuri. “LiSHT: Non-parametric Linearly Scaled Hyperbolic Tangent Activation Function for Neural Networks.” In: *Communications in Computer and Information Science*. Springer Nature Switzerland, 2023, pp. 462–476. DOI: 10.1007/978-3-031-31407-0_35. URL: https://doi.org/10.1007/978-3-031-31407-0_35 (cit. on p. 13).
- [192] A. Go, R. Bhayani, and L. Huang. *Twitter Sentiment Classification using Distant Supervision*. Tech. rep. 2009. URL: <https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf> (cit. on p. 13).
- [193] T. Sahni, C. Chandak, N. R. Chedeti, and M. Singh. “Efficient Twitter sentiment classification using subjective distant supervision.” In: *2017 9th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, Jan. 2017. DOI: 10.1109/comsnets.2017.7945451. URL: <https://doi.org/10.1109/comsnets.2017.7945451> (cit. on p. 13).
- [194] I. Vallés-Pérez, E. Soria-Olivas, M. Martínez-Sober, A. J. Serrano-López, J. Vila-Francés, and J. Gómez-Sanchís. “Empirical study of the modulus as activation function in computer vision applications.” In: *Engineering Applications of Artificial Intelligence* 120 (Apr. 2023), p. 105863. DOI: 10.1016/j.engappai.2023.105863. URL: <https://doi.org/10.1016/j.engappai.2023.105863> (cit. on pp. 13, 20, 21).
- [195] D. Misra. “Mish: A Self Regularized Non-Monotonic Activation Function.” In: *The 31st British Machine Vision Conference*. BMVC, Sept. 2020. URL: https://www.bmvc2020-conference.com/conference/papers/paper_0928.html (cit. on p. 13).
- [196] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition.” In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.90. URL: <https://doi.org/10.1109/cvpr.2016.90> (cit. on pp. 13, 47, 48, 50, 68).
- [197] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions.” In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015. DOI: 10.1109/cvpr.2015.7298594. URL: <https://doi.org/10.1109/cvpr.2015.7298594> (cit. on p. 13).
- [198] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. “Densely Connected Convolutional Networks.” In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.243. URL: <https://doi.org/10.1109/cvpr.2017.243> (cit. on pp. 13, 68, 70).
- [199] J. Hu, L. Shen, and G. Sun. “Squeeze-and-Excitation Networks.” In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00745. URL: <https://doi.org/10.1109/cvpr.2018.00745> (cit. on p. 13).
- [200] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. DOI: 10.48550/ARXIV.2004.10934. URL: <https://arxiv.org/abs/2004.10934> (cit. on p. 13).
- [201] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. “Scaled-YOLOv4: Scaling Cross Stage Partial Network.” In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. DOI: 10.1109/cvpr46437.2021.01283. URL: <https://doi.org/10.1109/cvpr46437.2021.01283> (cit. on pp. 13, 14).
- [202] X. Wang, H. Ren, and A. Wang. “Smish: A Novel Activation Function for Deep Learning Methods.” In: *Electronics* 11.4 (Feb. 2022), p. 540. ISSN: 2079-9292. DOI: 10.3390/electronics11040540. URL: <http://dx.doi.org/10.3390/electronics11040540> (cit. on p. 13).
- [203] X. Liu and X. Di. “TanhExp: A smooth activation function with high convergence speed for lightweight neural networks.” In: *IET Computer Vision* 15.2 (Feb. 2021), pp. 136–150. ISSN: 1751-9640. DOI: 10.1049/cvi2.12020. URL: <http://dx.doi.org/10.1049/cvi2.12020> (cit. on p. 13).
- [204] M. A. Mercioni and S. Holban. “TeLU: A New Activation Function for Deep Learning.” In: *2020 International Symposium on Electronics and Telecommunications (ISETC)*. IEEE, Nov. 2020. DOI: 10.1109/isetc50328.2020.9301084. URL: <http://dx.doi.org/10.1109/ISETC50328.2020.9301084> (cit. on pp. 13, 47).
- [205] S. Nag, M. Bhattacharyya, A. Mukherjee, and R. Kundu. “Serf: Towards better training of deep neural networks using log-Softplus EError activation Function.” In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Jan. 2023. DOI: 10.1109/wacv56688.2023.00529. URL: <http://dx.doi.org/10.1109/WACV56688.2023.00529> (cit. on p. 13).
- [206] D. Elliott, S. Frank, K. Sima'an, and L. Specia. “Multi30K: Multilingual English-German Image Descriptions.” In: *Proceedings of the 5th Workshop on Vision and Language*. Association for Computational Linguistics, 2016. DOI: 10.18653/v1/w16-3210. URL: <http://dx.doi.org/10.18653/v1/W16-3210> (cit. on p. 13).
- [207] E. Chai, W. Yu, T. Cui, J. Ren, and S. Ding. “An Efficient Asymmetric Nonlinear Activation Function for Deep Neural Networks.” In: *Symmetry* 14.5 (May 2022), p. 1027. DOI: 10.3390/sym14051027. URL: <https://doi.org/10.3390/sym14051027> (cit. on pp. 13, 14).
- [208] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. “Focal Loss for Dense Object Detection.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (Feb. 2020), pp. 318–327. DOI: 10.1109/tpami.2018.2858826. URL: <https://doi.org/10.1109/tpami.2018.2858826> (cit. on p. 14).
- [209] K. Douge, A. Berrahou, Y. T. Alaoui, and M. T. Alaoui. “A Self-gated Activation Function SINSIG Based on the Sine Trigonometric for Neural Network Models.” In: *Machine Learning for Networking*. Springer International Publishing, 2021, pp. 237–244. DOI: 10.1007/978-3-030-70866-5_15. URL: https://doi.org/10.1007/978-3-030-70866-5_15 (cit. on p. 14).
- [210] K. He, X. Zhang, S. Ren, and J. Sun. “Identity Mappings in Deep Residual Networks.” In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 630–645. DOI: 10.1007/978-3-319-46493-0_38. URL: https://doi.org/10.1007/978-3-319-46493-0_38 (cit. on p. 14).
- [211] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size*. 2016. DOI: 10.48550/ARXIV.1602.07360. URL: <https://arxiv.org/abs/1602.07360> (cit. on p. 14).
- [212] X. Zhang, X. Zhou, M. Lin, and J. Sun. “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.” In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00716. URL: <https://doi.org/10.1109/cvpr.2018.00716> (cit. on p. 14).
- [213] B. Ahmed, M. A. Haque, M. A. Iqbal, S. Jaiswal, U. B. Angadi, D. Kumar, and A. Rai. “DeepAProt: Deep learning based abiotic stress protein sequence classification and identification tool in cereals.” In: *Frontiers in Plant Science* 13 (Jan. 2023). ISSN: 1664-462X. DOI: 10.3389/fpls.2022.1008756. URL: <http://dx.doi.org/10.3389/fpls.2022.1008756> (cit. on p. 14).

- [214] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. “Language Modeling with Gated Convolutional Networks.” In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 933–941. URL: <https://proceedings.mlr.press/v70/dauphin17a.html> (cit. on p. 14).
- [215] N. Shazeer. *GLU Variants Improve Transformer*. 2020. DOI: 10.48550/ARXIV.2002.05202. URL: <https://arxiv.org/abs/2002.05202> (cit. on pp. 14, 15).
- [216] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. “Convolutional Sequence to Sequence Learning.” In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1243–1252. URL: <https://proceedings.mlr.press/v70/gehring17a.html> (cit. on p. 14).
- [217] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. “Conditional Image Generation with PixelCNN Decoders.” In: *Proceedings of the 30th International Conference on Neural Information Processing Systems. NIPS’16*. Barcelona, Spain: Curran Associates Inc., 2016, pp. 4797–4805. ISBN: 9781510838819. URL: <https://dl.acm.org/doi/10.5555/3157382.3157633> (cit. on p. 14).
- [218] J. Bridle. “Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters.” In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1989. URL: https://proceedings.neurips.cc/paper_files/paper/1989/file/0336dcbab05b9d5ad24f4333c7658a0e-Paper.pdf (cit. on p. 15).
- [219] T. Pearce, A. Brintrup, and J. Zhu. *Understanding Softmax Confidence and Uncertainty*. 2021. DOI: 10.48550/ARXIV.2106.04972. URL: <https://arxiv.org/abs/2106.04972> (cit. on p. 15).
- [220] M. Bhuvaneshwari and E. G. M. Kanaga. “Convolutional Neural Network for Addiction Detection using Improved Activation Function.” In: *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, Apr. 2021. DOI: 10.1109/iccmc51019.2021.9418022. URL: <http://dx.doi.org/10.1109/ICCMC51019.2021.9418022> (cit. on pp. 15, 58).
- [221] V. Nair and G. E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines.” In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Ed. by J. Fürnkranz and T. Joachims. Omnipress, 2010, pp. 807–814. URL: <http://www.cs.toronto.edu/~fritz/absps/reluICML.pdf> (cit. on pp. 15, 17, 42, 85).
- [222] X. Glorot, A. Bordes, and Y. Bengio. “Deep Sparse Rectifier Neural Networks.” In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by G. Gordon, D. Dunson, and M. Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 315–323. URL: <http://proceedings.mlr.press/v15/glorot11a.html> (cit. on pp. 15, 33).
- [223] A. Karpathy. *CS231n: Convolutional Neural Networks for Visual Recognition — Module 2: Convolutional Neural Networks*. 2023. URL: <https://cs231n.github.io/> (visited on 10/11/2023) (cit. on p. 15).
- [224] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks.” In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. Burges, L. Bottou, and K. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on p. 15).
- [225] K. Hara, D. Saito, and H. Shouno. “Analysis of function of rectified linear unit used in deep learning.” In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2015. DOI: 10.1109/ijcnn.2015.7280578. URL: <https://doi.org/10.1109/ijcnn.2015.7280578> (cit. on p. 15).
- [226] M. Telgarsky. “Neural Networks and Rational Functions.” In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 3387–3393. URL: <https://proceedings.mlr.press/v70/telgarsky17a.html> (cit. on p. 15).
- [227] A. Aleksandrov and K. Völlinger. “Formalizing Piecewise Affine Activation Functions of Neural Networks in Coq.” In: *NASA Formal Methods*. Springer Nature Switzerland, 2023, pp. 62–78. ISBN: 9783031331701. DOI: 10.1007/978-3-031-33170-1_4. URL: http://dx.doi.org/10.1007/978-3-031-33170-1_4 (cit. on p. 15).
- [228] D. Mishkin, N. Sergievskiy, and J. Matas. “Systematic evaluation of convolution neural network advances on the Imagenet.” In: *Computer Vision and Image Understanding* 161 (Aug. 2017), pp. 11–19. DOI: 10.1016/j.cviu.2017.05.007. URL: <https://doi.org/10.1016/j.cviu.2017.05.007> (cit. on p. 15).
- [229] W. Wang, X. Yang, B. C. Ooi, D. Zhang, and Y. Zhuang. “Effective deep learning-based multi-modal retrieval.” In: *The VLDB Journal* 25.1 (July 2015), pp. 79–101. DOI: 10.1007/s00778-015-0391-4. URL: <https://doi.org/10.1007/s00778-015-0391-4> (cit. on p. 15).
- [230] D. Mishkin and J. Matas. “All you need is a good init.” In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2016. URL: <http://arxiv.org/abs/1511.06422> (cit. on pp. 15, 16, 50, 73).
- [231] A. L. Maas, A. Y. Hannun, and A. Y. Ng. “Rectifier nonlinearities improve neural network acoustic models.” In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2013. URL: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf (cit. on pp. 15, 16, 40).
- [232] B. Graham. *Spatially-sparse convolutional neural networks*. 2014. DOI: 10.48550/ARXIV.1409.6070. URL: <https://arxiv.org/abs/1409.6070> (cit. on pp. 15, 16).
- [233] K. He, X. Zhang, S. Ren, and J. Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.” In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec. 2015. DOI: 10.1109/iccv.2015.123. URL: <https://doi.org/10.1109/iccv.2015.123> (cit. on pp. 15, 39, 40, 50, 53, 85).
- [234] B. Xu, N. Wang, T. Chen, and M. Li. *Empirical Evaluation of Rectified Activations in Convolutional Network*. 2015. DOI: 10.48550/ARXIV.1505.00853. URL: <https://arxiv.org/abs/1505.00853> (cit. on pp. 15, 16).
- [235] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan. “Deep Learning with S-Shaped Rectified Linear Activation Units.” In: *Proceedings of the AAAI Conference on Artificial Intelligence* 30.1 (Feb. 2016). DOI: 10.1609/aaai.v30i1.10287. URL: <https://doi.org/10.1609/aaai.v30i1.10287> (cit. on pp. 15, 67–69, 73).
- [236] D. Clevert, T. Unterthiner, and S. Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs).” In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2016. URL: <http://arxiv.org/abs/1511.07289> (cit. on pp. 15, 25, 43).
- [237] R. Parhi and R. D. Nowak. “The Role of Neural Network Activation Functions.” In: *IEEE Signal Processing Letters* 27 (2020), pp. 1779–1783. DOI: 10.1109/lsp.2020.3027517. URL: <https://doi.org/10.1109/lsp.2020.3027517> (cit. on p. 16).
- [238] B. H. Nayef, S. N. H. S. Abdullah, R. Sulaiman, and Z. A. A. Alyasseri. “Optimized leaky ReLU for handwritten Arabic character recognition using convolution neural networks.” In: *Multimedia Tools and Applications* 81.2 (Oct. 2021), pp. 2065–2094. DOI: 10.1007/s11042-021-11593-6. URL: <https://doi.org/10.1007/s11042-021-11593-6> (cit. on p. 16).
- [239] N. Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1929–1958. ISSN: 1532-4435. URL: <https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf> (cit. on p. 16).
- [240] M. K. Elakkiya and Deje. “Novel deep learning models with novel integrated activation functions for autism screening: AutoNet and MinAutoNet.” In: *Expert Systems with Applications* 238 (Mar. 2024), p. 122102. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.122102. URL: <http://dx.doi.org/10.1016/j.eswa.2023.122102> (cit. on pp. 16, 22).
- [241] M. K. Elakkiya and Deje. “Stacked autoencoder with novel integrated activation functions for the diagnosis of autism spectrum disorder.” In: *Neural Computing and Applications* 35.23 (Apr. 2023), pp. 17043–17075. ISSN: 1433-3058. DOI: 10.1007/s00521-023-08565-2. URL: <http://dx.doi.org/10.1007/s00521-023-08565-2> (cit. on p. 16).

- [242] J. Seo, J. Lee, and K. Kim. "Activation functions of deep neural networks for polar decoding applications." In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. IEEE, Oct. 2017. DOI: 10.1109/pimrc.2017.8292678. URL: <https://doi.org/10.1109/pimrc.2017.8292678> (cit. on p. 17).
- [243] A. Gupta and S. Ahuja. *Parametric Variational Linear Units (PVLUs) in Deep Convolutional Networks*. 2021. DOI: 10.48550/ARXIV.2110.12246. URL: <https://arxiv.org/abs/2110.12246> (cit. on pp. 17, 53).
- [244] W. Rodrigues. *SineReLU — An Alternative to the ReLU Activation Function*. 2018. URL: <https://wilder-rodrigues.medium.com/sinerelu-an-alternative-to-the-relu-activation-function-e46a6199997d> (visited on 01/28/2024) (cit. on p. 17).
- [245] K. Yamamichi and X.-H. Han. "Lightweight Multi-Scale Context Aggregation Deraining Network With Artifact-Attenuating Pooling and Activation Functions." In: *IEEE Access* 9 (2021), pp. 146948–146958. ISSN: 2169-3536. DOI: 10.1109/access.2021.3122450. URL: <http://dx.doi.org/10.1109/ACCESS.2021.3122450> (cit. on p. 17).
- [246] J. Cao, Y. Pang, X. Li, and J. Liang. "Randomly translational activation inspired by the input distributions of ReLU." In: *Neurocomputing* 275 (Jan. 2018), pp. 859–868. DOI: 10.1016/j.neucom.2017.09.031. URL: <https://doi.org/10.1016/j.neucom.2017.09.031> (cit. on pp. 17, 42).
- [247] Y. Liu, J. Zhang, C. Gao, J. Qu, and L. Ji. "Natural-Logarithm-Rectified Activation Function in Convolutional Neural Networks." In: *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*. IEEE, Dec. 2019. DOI: 10.1109/iccc47050.2019.9064398. URL: <https://doi.org/10.1109/iccc47050.2019.9064398> (cit. on p. 18).
- [248] H. Zhao, F. Liu, L. Li, and C. Luo. "A novel softplus linear unit for deep convolutional neural networks." In: *Applied Intelligence* 48.7 (Sept. 2017), pp. 1707–1720. DOI: 10.1007/s10489-017-1028-7. URL: <https://doi.org/10.1007/s10489-017-1028-7> (cit. on p. 18).
- [249] C. Xu, J. Huang, S.-p. Wang, and A.-q. Hu. "A Novel Parameterized Activation Function in Visual Geometry Group." In: *2018 2nd International Conference on Data Science and Business Analytics (ICDSBA)*. IEEE, Sept. 2018. DOI: 10.1109/icdsba.2018.00079. URL: <https://doi.org/10.1109/icdsba.2018.00079> (cit. on p. 18).
- [250] I. E. Jaafari, A. Ellahyani, and S. Charfi. "Parametric rectified nonlinear unit (PRenu) for convolution neural networks." In: *Signal, Image and Video Processing* 15.2 (July 2020), pp. 241–246. DOI: 10.1007/s11760-020-01746-9. URL: <https://doi.org/10.1007/s11760-020-01746-9> (cit. on p. 18).
- [251] S. S. Liew, M. Khalil-Hani, and R. Bakhteri. "Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems." In: *Neurocomputing* 216 (Dec. 2016), pp. 718–734. DOI: 10.1016/j.neucom.2016.08.037. URL: <https://doi.org/10.1016/j.neucom.2016.08.037> (cit. on pp. 18, 20, 34, 69).
- [252] E. Pishchik. "Trainable Activations for Image Classification." In: (Jan. 2023). DOI: 10.20944/preprints202301.0463.v1. URL: <https://doi.org/10.20944/preprints202301.0463.v1> (cit. on pp. 18, 26, 43, 48, 49, 61, 65, 73, 74).
- [253] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. *Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1*. 2016. DOI: 10.48550/ARXIV.1602.02830. URL: <https://arxiv.org/abs/1602.02830> (cit. on pp. 18, 19, 29).
- [254] R. Avenash and P. Viswanath. "Semantic Segmentation of Satellite Images using a Modified CNN with Hard-Swish Activation Function." In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2019. DOI: 10.5220/0007469604130420. URL: <http://dx.doi.org/10.5220/0007469604130420> (cit. on pp. 19, 68).
- [255] S. M. Waseem, A. V. Suraj, and S. K. Roy. "Accelerating the Activation Function Selection for Hybrid Deep Neural Networks – FPGA Implementation." In: *2021 IEEE Region 10 Symposium (TENSYP)*. IEEE, Aug. 2021. DOI: 10.1109/tensymp52854.2021.9551000. URL: <https://doi.org/10.1109/tensymp52854.2021.9551000> (cit. on p. 19).
- [256] D. Lupu and I. Necoara. "Exact Representation and Efficient Approximations of Linear Model Predictive Control Laws Via Hardtanh Type Deep Neural Networks." In: (2023). DOI: 10.2139/ssrn.4516044. URL: <http://dx.doi.org/10.2139/ssrn.4516044> (cit. on p. 19).
- [257] Z. Liu, H. Zhang, Z. Su, and X. Zhu. "Adaptive Binarization Method for Binary Neural Network." In: *2021 40th Chinese Control Conference (CCC)*. IEEE, July 2021. DOI: 10.23919/ccc52363.2021.9549344. URL: <http://dx.doi.org/10.23919/CCC52363.2021.9549344> (cit. on pp. 19, 44).
- [258] H. Kim, J. Park, C. Lee, and J.-J. Kim. "Improving Accuracy of Binary Neural Networks using Unbalanced Activation Distribution." In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. DOI: 10.1109/cvpr46437.2021.00777. URL: <http://dx.doi.org/10.1109/cvpr46437.2021.00777> (cit. on p. 19).
- [259] A. Howard et al. "Searching for MobileNetV3." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00140. URL: <http://dx.doi.org/10.1109/ICCV.2019.00140> (cit. on p. 19).
- [260] K. R. Konda, R. Memisevic, and D. Krueger. "Zero-bias autoencoders and the benefits of co-adapting features." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2015. URL: <http://arxiv.org/abs/1402.3337> (cit. on p. 20).
- [261] R. Goroshin and Y. LeCun. *Saturating Auto-Encoders*. 2013. DOI: 10.48550/ARXIV.1301.3577. URL: <https://arxiv.org/abs/1301.3577> (cit. on p. 20).
- [262] N. A. Golilarz and H. Demirel. "Thresholding neural network (TNN) based noise reduction with a new improved thresholding function." In: *Computational Research Progress in Applied Science & Engineering* 3.2 (2017), pp. 81–84. URL: https://jms.procedia.org/archive/CRPASE_169/CRPASE_proceedia_2017_3_2_3.pdf (cit. on p. 20).
- [263] R. Zhang, J. Yi, H. Tang, J. Xiang, and Y. Ren. "Fault Diagnosis Method of Waterproof Valves in Engineering Mixing Machinery Based on a New Adaptive Feature Extraction Model." In: *Energies* 15.8 (Apr. 2022), p. 2796. ISSN: 1996-1073. DOI: 10.3390/en15082796. URL: <http://dx.doi.org/10.3390/en15082796> (cit. on p. 20).
- [264] H. Hu. "vReLU Activation Functions for Artificial Neural Networks." In: *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE, July 2018. DOI: 10.1109/fskd.2018.8687140. URL: <https://doi.org/10.1109/fskd.2018.8687140> (cit. on p. 20).
- [265] H. Hu. "Symmetric Rectified Linear Units for Fully Connected Deep Models." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2018, pp. 291–298. ISBN: 9783319992471. DOI: 10.1007/978-3-319-99247-1_26. URL: http://dx.doi.org/10.1007/978-3-319-99247-1_26 (cit. on p. 20).
- [266] O. I. Bergardt. *Improving Classification Neural Networks by using Absolute activation function (MNIST/LeNET-5 example)*. 2023. DOI: 10.48550/ARXIV.2304.11758. URL: <https://arxiv.org/abs/2304.11758> (cit. on p. 20).
- [267] T. Y. Pan, G. Yang, J. Zhao, and J. Ding. "Smoothing piecewise linear activation functions based on mollified square root functions." In: *Mathematical Foundations of Computing* (2023). ISSN: 2577-8838. DOI: 10.3934/mfc.2023032. URL: <http://dx.doi.org/10.3934/mfc.2023032> (cit. on pp. 20, 21, 70, 78–81).
- [268] X. Bresson, S. Eshedoglu, P. Vanderghenst, J.-P. Thiran, and S. Osher. "Fast Global Minimization of the Active Contour/Snake Model." In: *Journal of Mathematical Imaging and Vision* 28.2 (July 2007), pp. 151–167. ISSN: 1573-7683. DOI: 10.1007/s10851-007-0002-0. URL: <http://dx.doi.org/10.1007/s10851-007-0002-0> (cit. on pp. 20, 78).
- [269] A. A. Alkhouly, A. Mohammed, and H. A. Hefny. "Improving the Performance of Deep Neural Networks Using Two Proposed Activation Functions." In: *IEEE Access* 9 (2021), pp. 82249–82271. ISSN: 2169-3536. DOI: 10.1109/access.2021.3085855. URL: <http://dx.doi.org/10.1109/ACCESS.2021.3085855> (cit. on pp. 21, 27).
- [270] Q. Zhao and L. D. Griffin. *Suppressing the Unusual: towards Robust CNNs using Symmetric Activation Functions*. 2016. DOI: 10.48550/ARXIV.1603.05145. URL: <https://arxiv.org/abs/1603.05145> (cit. on p. 21).

- [271] C.-H. Shan, X.-R. Guo, and J. Ou. "Deep Leaky Single-peaked Triangle Neural Networks." In: *International Journal of Control, Automation and Systems* 17.10 (Aug. 2019), pp. 2693–2701. ISSN: 2005-4092. DOI: 10.1007/s12555-018-0796-0. URL: <http://dx.doi.org/10.1007/s12555-018-0796-0> (cit. on p. 21).
- [272] A. S. D. Desabathula and S. Eluri. "A novel Leaky Rectified Triangle Linear Unit based Deep Convolutional Neural Network for facial emotion recognition." In: *Multimedia Tools and Applications* 82.12 (Nov. 2022), pp. 18669–18689. ISSN: 1573-7721. DOI: 10.1007/s11042-022-14186-z. URL: <http://dx.doi.org/10.1007/s11042-022-14186-z> (cit. on p. 21).
- [273] G. Lin and W. Shen. "Research on convolutional neural network based on improved Relu piecewise activation function." In: *Procedia Computer Science* 131 (2018), pp. 977–984. DOI: 10.1016/j.procs.2018.04.239. URL: <https://doi.org/10.1016/j.procs.2018.04.239> (cit. on p. 21).
- [274] J. Li, H. Feng, and D.-X. Zhou. "SignReLU neural network and its approximation ability." In: *Journal of Computational and Applied Mathematics* 440 (Apr. 2024), p. 115551. ISSN: 0377-0427. DOI: 10.1016/j.cam.2023.115551. URL: <http://dx.doi.org/10.1016/j.cam.2023.115551> (cit. on p. 21).
- [275] J. Li, H. Feng, and D.-X. Zhou. *A new activation for neural networks and its approximation*. 2022. URL: <https://arxiv.org/abs/2210.10264v1> (cit. on p. 21).
- [276] W. Shang, K. Sohn, D. Almeida, and H. Lee. "Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2217–2225. URL: <https://proceedings.mlr.press/v48/shang16.html> (cit. on p. 22).
- [277] S. Zagoruyko and N. Komodakis. *DiracNets: Training Very Deep Neural Networks Without Skip-Connections*. 2017. DOI: 10.48550/ARXIV.1706.00388. URL: <https://arxiv.org/abs/1706.00388> (cit. on p. 22).
- [278] L. Eidnes and A. Nøklund. *Shifting Mean Activation Towards Zero with Bipolar Activation Functions*. 2017. DOI: 10.48550/ARXIV.1709.04054. URL: <https://arxiv.org/abs/1709.04054> (cit. on p. 22).
- [279] F. Godin, J. Degraeve, J. Dambre, and W. D. Neve. "Dual Rectified Linear Units (DReLU): A replacement for tanh activation functions in Quasi-Recurrent Neural Networks." In: *Pattern Recognition Letters* 116 (Dec. 2018), pp. 8–14. DOI: 10.1016/j.patrec.2018.09.006. URL: <https://doi.org/10.1016/j.patrec.2018.09.006> (cit. on pp. 22, 26).
- [280] A. Chernodub and D. Nowicki. *Norm-preserving Orthogonal Permutation Linear Unit Activation Functions (OPLU)*. 2016. DOI: 10.48550/ARXIV.1604.02313. URL: <https://arxiv.org/abs/1604.02313> (cit. on p. 22).
- [281] X. Jiang, Y. Pang, X. Li, J. Pan, and Y. Xie. "Deep neural networks with Elastic Rectified Linear Units for object recognition." In: *Neurocomputing* 275 (Jan. 2018), pp. 1132–1139. DOI: 10.1016/j.neucom.2017.09.056. URL: <https://doi.org/10.1016/j.neucom.2017.09.056> (cit. on pp. 23, 45).
- [282] Y. Berradi. "Symmetric Power Activation Functions for Deep Neural Networks." In: *Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications*. ACM, May 2018. DOI: 10.1145/3230905.3230956. URL: <https://doi.org/10.1145/3230905.3230956> (cit. on p. 23).
- [283] B. Li, S. Tang, and H. Yu. "PowerNet: Efficient Representations of Polynomials and Smooth Functions by Deep Neural Networks with Rectified Power Units." In: (2019). DOI: 10.48550/ARXIV.1909.05136. URL: <https://arxiv.org/abs/1909.05136> (cit. on p. 23).
- [284] T. J. Heeringa, L. Spek, F. Schwenninger, and C. Brune. *Embeddings between Barron spaces with higher order activation functions*. 2023. DOI: 10.48550/ARXIV.2305.15839. URL: <https://arxiv.org/abs/2305.15839> (cit. on p. 23).
- [285] D. R. So, W. Mañke, H. Liu, Z. Dai, N. Shazeer, and Q. V. Le. *Primer: Searching for Efficient Transformers for Language Modeling*. 2021. DOI: 10.48550/ARXIV.2109.08668. URL: <https://arxiv.org/abs/2109.08668> (cit. on p. 23).
- [286] S. Saha, N. Nagaraj, A. Mathur, R. Yedida, and S. H. R. "Evolution of novel activation functions in neural network training for astronomy data: habitability classification of exoplanets." In: *The European Physical Journal Special Topics* 229.16 (Nov. 2020), pp. 2629–2738. DOI: 10.1140/epjst/e2020-000098-9. URL: <https://doi.org/10.1140/epjst/e2020-000098-9> (cit. on pp. 23, 32).
- [287] I. Mediratta, S. Saha, and S. Mathur. "LipARELU: ARELU Networks aided by Lipschitz Acceleration." In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2021. DOI: 10.1109/ijcnn52387.2021.9533853. URL: <http://dx.doi.org/10.1109/ijcnn52387.2021.9533853> (cit. on p. 23).
- [288] K. Nasiri and K. Ghiasi-Shirazi. "PowerLinear Activation Functions with application to the first layer of CNNs." In: *2021 11th International Conference on Computer Engineering and Knowledge (ICCKE)*. IEEE, Oct. 2021. DOI: 10.1109/iccke54056.2021.9721450. URL: <http://dx.doi.org/10.1109/ICCKE54056.2021.9721450> (cit. on pp. 23, 24).
- [289] S. R. Dubey and S. Chakraborty. "Average biased ReLU based CNN descriptor for improved face retrieval." In: *Multimedia Tools and Applications* 80.15 (Jan. 2021), pp. 23181–23206. DOI: 10.1007/s11042-020-10269-x. URL: <https://doi.org/10.1007/s11042-020-10269-x> (cit. on p. 24).
- [290] C. Shan, A. Li, and X. Chen. "Deep delay rectified neural networks." In: *The Journal of Supercomputing* 79.1 (July 2022), pp. 880–896. ISSN: 1573-0484. DOI: 10.1007/s11227-022-04704-z. URL: <http://dx.doi.org/10.1007/s11227-022-04704-z> (cit. on p. 24).
- [291] D. Macêdo, C. Zanchettin, A. L. Oliveira, and T. Ludermir. "Enhancing batch normalized convolutional networks using displaced rectifier linear units: A systematic comparative study." In: *Expert Systems with Applications* 124 (June 2019), pp. 271–281. DOI: 10.1016/j.eswa.2019.01.066. URL: <https://doi.org/10.1016/j.eswa.2019.01.066> (cit. on p. 24).
- [292] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *CoRR* abs/1409.1556 (2014). URL: <http://arxiv.org/abs/1409.1556> (cit. on p. 24).
- [293] H. Yang, A. Alsadoon, P. W. C. Prasad, T. Al-Dala'in, T. A. Rashid, A. Maag, and O. H. Alsadoon. "Deep learning neural networks for emotion classification from text: enhanced leaky rectified linear unit activation and weighted loss." In: *Multimedia Tools and Applications* 81.11 (Feb. 2022), pp. 15439–15468. ISSN: 1573-7721. DOI: 10.1007/s11042-022-12629-1. URL: <http://dx.doi.org/10.1007/s11042-022-12629-1> (cit. on p. 24).
- [294] H. H. Chieng, N. Wahid, O. Pauline, and S. R. K. Perla. "Flatten-T Swish: a thresholded ReLU-Swish-like activation function for deep learning." In: *International Journal of Advances in Intelligent Informatics* 4.2 (July 2018), p. 76. DOI: 10.26555/ijain.v4i2.249. URL: <https://doi.org/10.26555/ijain.v4i2.249> (cit. on p. 25).
- [295] O. Sharma. "A Novel Activation Function in Convolutional Neural Network for Image Classification in Deep Learning." In: *Data Science and Analytics*. Springer Singapore, 2020, pp. 120–130. DOI: 10.1007/978-981-15-5827-6_10. URL: https://doi.org/10.1007/978-981-15-5827-6_10 (cit. on p. 25).
- [296] Y. Ying, J. Su, P. Shan, L. Miao, X. Wang, and S. Peng. "Rectified Exponential Units for Convolutional Neural Networks." In: *IEEE Access* 7 (2019), pp. 101633–101640. DOI: 10.1109/access.2019.2928442. URL: <https://doi.org/10.1109/access.2019.2928442> (cit. on pp. 25, 42).
- [297] M.-I. Georgescu, R. T. Ionescu, N.-C. Ristea, and N. Sebe. "Nonlinear neurons with human-like apical dendrite activations." In: *Applied Intelligence* 53.21 (Aug. 2023), pp. 25984–26007. ISSN: 1573-7497. DOI: 10.1007/s10489-023-04921-w. URL: <http://dx.doi.org/10.1007/s10489-023-04921-w> (cit. on p. 25).
- [298] M. N. Qureshi and M. Sarosh Umar. "SaRa: A Novel Activation Function with Application to Melanoma Image Classification." In: *2022 International Conference on Automation, Computing and Renewable Systems (ICACRS)*. IEEE, Dec. 2022. DOI: 10.1109/icacrs55517.2022.10029161. URL: <http://dx.doi.org/10.1109/ICACRS55517.2022.10029161> (cit. on p. 26).
- [299] H. Fu, H. Shi, Y. Xu, and J. Shao. "Research on Gas Outburst Prediction Model Based on Multiple Strategy Fusion Improved Snake Optimization Algorithm With Temporal Convolutional Network." In: *IEEE Access* 10 (2022), pp. 117973–117984. ISSN: 2169-3536. DOI: 10.1109/access.2022.3220765. URL: <http://dx.doi.org/10.1109/ACCESS.2022.3220765> (cit. on p. 26).

- [300] Z. Hu, H. Huang, Q. Ran, and M. Yuan. “Improving Convolutional Neural Network Expression via Difference Exponentially Linear Units.” In: *Journal of Physics: Conference Series* 1651.1 (Nov. 2020), p. 012163. ISSN: 1742-6596. DOI: 10.1088/1742-6596/1651/1/012163. URL: <http://dx.doi.org/10.1088/1742-6596/1651/1/012163> (cit. on p. 26).
- [301] H.-S. Feng and C.-H. Yang. “PolyLU: A Simple and Robust Polynomial-Based Linear Unit Activation Function for Deep Learning.” In: *IEEE Access* 11 (2023), pp. 101347–101358. ISSN: 2169-3536. DOI: 10.1109/access.2023.3315308. URL: <http://dx.doi.org/10.1109/ACCESS.2023.3315308> (cit. on p. 27).
- [302] Kaggle. *Dogs vs. Cats*. 2013. URL: <https://www.kaggle.com/c/dogs-vs-cats/data> (visited on 01/12/2024) (cit. on p. 27).
- [303] J. Elson, J. R. Douceur, J. Howell, and J. Saul. “Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization.” In: *Proceedings of the 14th ACM conference on Computer and communications security*. CCS07. ACM, Oct. 2007. DOI: 10.1145/1315245.1315291. URL: <http://dx.doi.org/10.1145/1315245.1315291> (cit. on p. 27).
- [304] B. Duan, Y. Yang, and X. Dai. “Activation by Switch Unit of Opposite First Powers.” In: *2022 IEEE 8th International Conference on Computer and Communications (ICCC)*. IEEE, Dec. 2022. DOI: 10.1109/iccc56324.2022.10065932. URL: <http://dx.doi.org/10.1109/ICCC56324.2022.10065932> (cit. on pp. 27, 53).
- [305] Y. Li, P. L. K. Ding, and B. Li. *Training Neural Networks by Using Power Linear Units (PoLUs)*. 2018. DOI: 10.48550/ARXIV.1802.00212. URL: <https://arxiv.org/abs/1802.00212> (cit. on p. 27).
- [306] M. Zhu, W. Min, Q. Wang, S. Zou, and X. Chen. “PFLU and FPFLU: Two novel non-monotonic activation functions in convolutional neural networks.” In: *Neurocomputing* 429 (Mar. 2021), pp. 110–117. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2020.11.068. URL: <http://dx.doi.org/10.1016/j.neucom.2020.11.068> (cit. on p. 27).
- [307] C. Zhang, Y. Xu, and Z. Sheng. “Elastic Adaptively Parametric Compounded Units for Convolutional Neural Network.” In: *Journal of Advanced Computational Intelligence and Intelligent Informatics* 27.4 (July 2023), pp. 576–584. ISSN: 1343-0130. DOI: 10.20965/jaciii.2023.p0576. URL: <http://dx.doi.org/10.20965/jaciii.2023.p0576> (cit. on p. 27).
- [308] M. Basirat and P. M. Roth. “L*ReLU: Piece-wise Linear Activation Functions for Deep Fine-grained Visual Categorization.” In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2020. DOI: 10.1109/wacv45572.2020.9093485. URL: <https://doi.org/10.1109/wacv45572.2020.9093485> (cit. on p. 28).
- [309] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. “Self-Normalizing Neural Networks.” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/5d44ee6f2c3f71b73125876103c8f6c4-Paper.pdf (cit. on pp. 28, 48).
- [310] Z. Chen, Z. WeiQin, L. Deng, G. Li, and Y. Xie. *Redefining The Self-Normalization Property*. 2021. URL: <https://openreview.net/forum?id=gfwf0skyzSx> (cit. on p. 28).
- [311] G. Zhang and H. Li. *Effectiveness of Scaled Exponentially-Regularized Linear Units (SERLUs)*. 2018. DOI: 10.48550/ARXIV.1807.10117. URL: <https://arxiv.org/abs/1807.10117> (cit. on p. 28).
- [312] S. Hermawan and R. Mandala. “Improving Accuracy using The ASERLU layer in CNN-BiLSTM Architecture on Sentiment Analysis.” In: *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)* 5.5 (Oct. 2021), pp. 1001–1007. ISSN: 2580-0760. DOI: 10.29207/resti.v5i5.3534. URL: <http://dx.doi.org/10.29207/resti.v5i5.3534> (cit. on p. 28).
- [313] S. Kiliçarslan and M. Celik. “RSigELU: A nonlinear activation function for deep neural networks.” In: *Expert Systems with Applications* 174 (July 2021), p. 114805. DOI: 10.1016/j.eswa.2021.114805. URL: <https://doi.org/10.1016/j.eswa.2021.114805> (cit. on p. 29).
- [314] H. Xiao, K. Rasul, and R. Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. 2017. DOI: 10.48550/ARXIV.1708.07747. URL: <https://arxiv.org/abs/1708.07747> (cit. on pp. 29, 63, 75).
- [315] S. Kiliçarslan. “A novel nonlinear hybrid HardSReLU activation function in transfer learning architectures for hemorrhage classification.” In: *Multimedia Tools and Applications* 82.4 (Dec. 2022), pp. 6345–6365. ISSN: 1573-7721. DOI: 10.1007/s11042-022-14313-w. URL: <http://dx.doi.org/10.1007/s11042-022-14313-w> (cit. on p. 29).
- [316] Y. Wang, Y. Li, Y. Song, and X. Rong. “The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition.” In: *Applied Sciences* 10.5 (Mar. 2020), p. 1897. ISSN: 2076-3417. DOI: 10.3390/app10051897. URL: <http://dx.doi.org/10.3390/app10051897> (cit. on p. 29).
- [317] A. Wuraola, N. Patel, and S. K. Nguang. “Efficient activation functions for embedded inference engines.” In: *Neurocomputing* 442 (June 2021), pp. 73–88. DOI: 10.1016/j.neucom.2021.02.030. URL: <https://doi.org/10.1016/j.neucom.2021.02.030> (cit. on pp. 30, 31).
- [318] A. Wuraola and N. Patel. “SQLN: A New Computationally Efficient Activation Function.” In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2018. DOI: 10.1109/ijcnn.2018.8489043. URL: <https://doi.org/10.1109/ijcnn.2018.8489043> (cit. on pp. 30, 31).
- [319] J. Bilski and A. I. Galushkin. “A New Proposition of the Activation Function for Significant Improvement of Neural Networks Performance.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 35–45. ISBN: 9783319393780. DOI: 10.1007/978-3-319-39378-0_4. URL: http://dx.doi.org/10.1007/978-3-319-39378-0_4 (cit. on pp. 30, 31).
- [320] D. Dua and E. Karra Taniskidou. *UCI Machine Learning Repository*. 2017. URL: <http://archive.ics.uci.edu/ml> (cit. on p. 30).
- [321] B. Carlile, G. Delamarter, P. Kinney, A. Marti, and B. Whitney. *Improving Deep Learning by Inverse Square Root Linear Units (ISRLUs)*. 2017. DOI: 10.48550/ARXIV.1710.09967. URL: <https://arxiv.org/abs/1710.09967> (cit. on p. 31).
- [322] X. Yang, Y. Chen, and H. Liang. “Square Root Based Activation Function in Neural Networks.” In: *2018 International Conference on Audio, Language and Image Processing (ICALIP)*. IEEE, July 2018. DOI: 10.1109/icalip.2018.8455590. URL: <https://doi.org/10.1109/icalip.2018.8455590> (cit. on p. 32).
- [323] M. Khachumov, Y. Emelyanova, and V. Khachumov. “Parabola-Based Artificial Neural Network Activation Functions.” In: *2023 International Russian Automation Conference (RusAutoCon)*. IEEE, Sept. 2023. DOI: 10.1109/rusautocon58002.2023.10272855. URL: <http://dx.doi.org/10.1109/RusAutoCon58002.2023.10272855> (cit. on p. 32).
- [324] Mate Labs. *Secret Sauce behind the beauty of Deep Learning: Beginners guide to Activation Functions*. 2017. URL: <https://towardsdatascience.com/secret-sauce-behind-the-beauty-of-deep-learning-beginners-guide-to-activation-functions-a8e23a57d046> (visited on 01/28/2024) (cit. on p. 32).
- [325] A. Mishra, P. Chandra, and U. Ghose. “A New Activation Function Validated on Function Approximation Tasks.” In: *Proceedings of 3rd International Conference on Computing Informatics and Networks*. Springer Singapore, 2021, pp. 311–321. ISBN: 9789811597121. DOI: 10.1007/978-981-15-9712-1_26. URL: http://dx.doi.org/10.1007/978-981-15-9712-1_26 (cit. on p. 32).
- [326] S. Saha, A. Mathur, K. Bora, S. Basak, and S. Agrawal. “A New Activation Function for Artificial Neural Net Based Habitability Classification.” In: *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, Sept. 2018. DOI: 10.1109/icacci.2018.8554460. URL: <http://dx.doi.org/10.1109/ICACCI.2018.8554460> (cit. on p. 32).
- [327] J. Bilski. “The Backpropagation learning with logarithmic transfer function.” In: *Proc. 5th Conf. On Neural Networks and Soft Computing*. 2000, pp. 71–76 (cit. on p. 32).
- [328] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. *Mastering Diverse Domains through World Models*. 2023. DOI: 10.48550/ARXIV.2301.04104. URL: <https://arxiv.org/abs/2301.04104> (cit. on pp. 32, 33).

- [329] J. Kiseľák, Y. Lu, J. Švihra, P. Szépe, and M. Stehlík. “SPOCU”: scaled polynomial constant unit activation function.” In: *Neural Computing and Applications* 33.8 (July 2020), pp. 3385–3401. ISSN: 1433-3058. DOI: 10.1007/s00521-020-05182-1. URL: <http://dx.doi.org/10.1007/s00521-020-05182-1> (cit. on p. 33).
- [330] J. Kiseľák, Y. Lu, J. Švihra, P. Szépe, and M. Stehlík. “Correction to: “SPOCU”: scaled polynomial constant unit activation function.” In: *Neural Computing and Applications* 33.5 (Nov. 2020), pp. 1749–1750. ISSN: 1433-3058. DOI: 10.1007/s00521-020-05412-6. URL: <http://dx.doi.org/10.1007/s00521-020-05412-6> (cit. on p. 33).
- [331] S.-Y. Hwang and J.-J. Kim. “A Universal Activation Function for Deep Learning.” In: *Computers, Materials & Continua* 75.2 (2023), pp. 3553–3569. ISSN: 1546-2226. DOI: 10.32604/cmc.2023.037028. URL: <http://dx.doi.org/10.32604/cmc.2023.037028> (cit. on p. 33).
- [332] B. Yuen, M. T. Hoang, X. Dong, and T. Lu. “Universal activation function for machine learning.” In: *Scientific Reports* 11.1 (Sept. 2021). ISSN: 2045-2322. DOI: 10.1038/s41598-021-96723-8. URL: <http://dx.doi.org/10.1038/s41598-021-96723-8> (cit. on pp. 33, 65).
- [333] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia. “Incorporating Second-Order Functional Knowledge for Better Option Pricing.” In: *Advances in Neural Information Processing Systems*. Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press, 2000. URL: https://proceedings.neurips.cc/paper_files/paper/2000/file/44968aece94f667e4095002d140b5896-Paper.pdf (cit. on p. 33).
- [334] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li. “Improving deep neural networks using softplus units.” In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2015. DOI: 10.1109/ijcnn.2015.7280459. URL: <https://doi.org/10.1109/ijcnn.2015.7280459> (cit. on p. 33).
- [335] Q. Liu and S. Furber. “Noisy Softplus: A Biology Inspired Activation Function.” In: *Neural Information Processing*. Springer International Publishing, 2016, pp. 405–412. DOI: 10.1007/978-3-319-46681-1_49. URL: https://doi.org/10.1007/978-3-319-46681-1_49 (cit. on p. 33).
- [336] K. Sun, J. Yu, L. Zhang, and Z. Dong. “A Convolutional Neural Network Model Based on Improved Softplus Activation Function.” In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, July 2019, pp. 1326–1335. DOI: 10.1007/978-3-030-25128-4_164. URL: https://doi.org/10.1007/978-3-030-25128-4_164 (cit. on p. 33).
- [337] A. Ciuparu, A. Nagy-Dăbăcan, and R. C. Mureșan. “Soft++, a multi-parametric non-saturating non-linearity that improves convergence in deep neural architectures.” In: *Neurocomputing* 384 (Apr. 2020), pp. 376–388. DOI: 10.1016/j.neucom.2019.12.014. URL: <https://doi.org/10.1016/j.neucom.2019.12.014> (cit. on p. 34).
- [338] Y. Chen, Y. Mai, J. Xiao, and L. Zhang. “Improving the Antinoise Ability of DNNs via a Bio-Inspired Noise Adaptive Activation Function Rand Softplus.” In: *Neural Computation* 31.6 (June 2019), pp. 1215–1233. DOI: 10.1162/neco_a_01192. URL: https://doi.org/10.1162/neco_a_01192 (cit. on p. 34).
- [339] G. S. da S. Gomes and T. B. Ludermit. “Optimization of the weights and asymmetric activation function family of neural network for time series forecasting.” In: *Expert Systems with Applications* 40.16 (Nov. 2013), pp. 6438–6446. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2013.05.053. URL: <http://dx.doi.org/10.1016/j.eswa.2013.05.053> (cit. on p. 34).
- [340] F. J. Aranda-Ordaz. “On two families of transformations to additivity for binary response data.” In: *Biometrika* 68.2 (1981), pp. 357–363. ISSN: 1464-3510. DOI: 10.1093/biomet/68.2.357. URL: <http://dx.doi.org/10.1093/biomet/68.2.357> (cit. on p. 34).
- [341] J.-C. Li, W. W. Y. Ng, D. S. Yeung, and P. P. K. Chan. “Bi-firing deep neural networks.” In: *International Journal of Machine Learning and Cybernetics* 5.1 (Sept. 2013), pp. 73–83. DOI: 10.1007/s13042-013-0198-9. URL: <https://doi.org/10.1007/s13042-013-0198-9> (cit. on p. 34).
- [342] P. Liu, Z. Zeng, and J. Wang. “Multistability analysis of a general class of recurrent neural networks with non-monotonic activation functions and time-varying delays.” In: *Neural Networks* 79 (July 2016), pp. 117–127. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2016.03.010. URL: <http://dx.doi.org/10.1016/j.neunet.2016.03.010> (cit. on pp. 34, 35).
- [343] L. Parisi. *m-arcsinh: An Efficient and Reliable Function for SVM and MLP in scikit-learn*. 2020. DOI: 10.48550/ARXIV.2009.07530. URL: <https://arxiv.org/abs/2009.07530> (cit. on p. 35).
- [344] L. Parisi, R. Ma, N. RaviChandran, and M. Lanzillotta. “hyper-sinh: An accurate and reliable function from shallow to deep learning in TensorFlow and Keras.” In: *Machine Learning with Applications* 6 (Dec. 2021), p. 100112. ISSN: 2666-8270. DOI: 10.1016/j.mlwa.2021.100112. URL: <http://dx.doi.org/10.1016/j.mlwa.2021.100112> (cit. on p. 35).
- [345] L. Parisi, A. Zaemia, R. Ma, and M. Youseffi. “Hyper-sinh-Convolutional Neural Network for Early Detection of Parkinson’s Disease from Spiral Drawings.” In: *WSEAS TRANSACTIONS ON COMPUTER RESEARCH* 9 (Mar. 2021), pp. 1–7. ISSN: 1991-8755. DOI: 10.37394/232018.2021.9.1. URL: <http://dx.doi.org/10.37394/232018.2021.9.1> (cit. on p. 35).
- [346] K. Hara and K. Nakayama. “Comparison of activation functions in multilayer neural network for pattern classification.” In: *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN’94)*. ICNN-94. IEEE, 1994. DOI: 10.1109/icnn.1994.374710. URL: <http://dx.doi.org/10.1109/icnn.1994.374710> (cit. on pp. 35–37).
- [347] M. S. Gashler and S. C. Ashmore. “Training Deep Fourier Neural Networks to Fit Time-Series Data.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 48–55. ISBN: 9783319093307. DOI: 10.1007/978-3-319-09330-7_7. URL: http://dx.doi.org/10.1007/978-3-319-09330-7_7 (cit. on p. 35).
- [348] Y. Zhang, L. Qu, J. Liu, D. Guo, and M. Li. “Sine neural network (SNN) with double-stage weights and structure determination (DS-WASD).” In: *Soft Computing* 20.1 (Oct. 2014), pp. 211–221. ISSN: 1433-7479. DOI: 10.1007/s00500-014-1491-6. URL: <http://dx.doi.org/10.1007/s00500-014-1491-6> (cit. on p. 35).
- [349] J. Sopena. “Neural networks with periodic and monotonic activation functions: a comparative study in classification problems.” In: *9th International Conference on Artificial Neural Networks: ICANN ’99*. IEE, 1999. DOI: 10.1049/cp:19991129. URL: <http://dx.doi.org/10.1049/cp:19991129> (cit. on p. 35).
- [350] S. A. Faroughi, R. Soltanmohammadi, P. Datta, S. K. Mahjour, and S. Faroughi. “Physics-Informed Neural Networks with Periodic Activation Functions for Solute Transport in Heterogeneous Porous Media.” In: *Mathematics* 12.1 (Dec. 2023), p. 63. ISSN: 2227-7390. DOI: 10.3390/math12010063. URL: <http://dx.doi.org/10.3390/math12010063> (cit. on p. 35).
- [351] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. “Implicit Neural Representations with Periodic Activation Functions.” In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 7462–7473. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf (cit. on p. 35).
- [352] W. X. Cheng, P. Suganthan, and R. Katuwal. “Time series classification using diversified Ensemble Deep Random Vector Functional Link and Resnet features.” In: *Applied Soft Computing* 112 (Nov. 2021), p. 107826. ISSN: 1568-4946. DOI: 10.1016/j.asoc.2021.107826. URL: <http://dx.doi.org/10.1016/j.asoc.2021.107826> (cit. on p. 35).
- [353] A. Daskin. “A Simple Quantum Neural Net with a Periodic Activation Function.” In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Oct. 2018. DOI: 10.1109/smc.2018.00491. URL: <http://dx.doi.org/10.1109/smc.2018.00491> (cit. on p. 35).
- [354] K.-H. Chan, S.-K. Im, W. Ke, and N.-L. Lei. “SinP[N]: A Fast Convergence Activation Function for Convolutional Neural Networks.” In: *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*. IEEE, Dec. 2018. DOI: 10.1109/ucc-companion.2018.00082. URL: <http://dx.doi.org/10.1109/UCC-Companion.2018.00082> (cit. on p. 36).
- [355] M. M. Noel, A. L. A. Trivedi, and P. Dutta. *Growing Cosine Unit: A Novel Oscillatory Activation Function That Can Speedup Training and Reduce Parameters in Convolutional Neural Networks*. 2021. DOI: 10.48550/ARXIV.2108.12943. URL: <https://arxiv.org/abs/2108.12943> (cit. on p. 36).
- [356] J. Sharma. *Evaluating CNN with Oscillatory Activation Function*. 2022. DOI: 10.48550/ARXIV.2211.06878. URL: <https://arxiv.org/abs/2211.06878> (cit. on p. 36).
- [357] P. Sharma, A. R. Sahoo, S. Sinha, and S. Bharadwaj. “NFT artwork generation using oscillatory activation functions in GANs.” In: (Mar. 2022). DOI: 10.31224/2225. URL: <http://dx.doi.org/10.31224/2225> (cit. on p. 36).

- [358] J. U. Rahman, F. Makhdoom, and D. Lu. *Amplifying Sine Unit: An Oscillatory Activation Function for Deep Neural Networks to Recover Nonlinear Oscillations Efficiently*. 2023. DOI: 10.48550/ARXIV.2304.09759. URL: <https://arxiv.org/abs/2304.09759> (cit. on p. 36).
- [359] J. U. Rahman, F. Makhdoom, and D. Lu. *ASU-CNN: An Efficient Deep Architecture for Image Classification and Feature Visualizations*. 2023. DOI: 10.48550/ARXIV.2305.19146. URL: <https://arxiv.org/abs/2305.19146> (cit. on p. 36).
- [360] M. Ö. Efe. “Novel Neuronal Activation Functions for Feedforward Neural Networks.” In: *Neural Processing Letters* 28.2 (Aug. 2008), pp. 63–79. ISSN: 1573-773X. DOI: 10.1007/s11063-008-9082-0. URL: <http://dx.doi.org/10.1007/s11063-008-9082-0> (cit. on pp. 36, 37, 71).
- [361] M. Gustineli. *A survey on recently proposed activation functions for Deep Learning*. 2022. DOI: 10.48550/ARXIV.2204.02921. URL: <https://arxiv.org/abs/2204.02921> (cit. on pp. 36, 37).
- [362] H. Abdel-Nabi, G. Al-Naymat, M. Z. Ali, and A. Awajan. “HcLSH: A Novel Non-Linear Monotonic Activation Function for Deep Learning Methods.” In: *IEEE Access* 11 (2023), pp. 47794–47815. ISSN: 2169-3536. DOI: 10.1109/access.2023.3276298. URL: <http://dx.doi.org/10.1109/ACCESS.2023.3276298> (cit. on p. 36).
- [363] T. Kalaiselvi, S. T. Padmapriya, K. Somasundaram, and S. Praveenkumar. “E-Tanh: a novel activation function for image processing neural network models.” In: *Neural Computing and Applications* 34.19 (June 2022), pp. 16563–16575. ISSN: 1433-3058. DOI: 10.1007/s00521-022-07245-x. URL: <http://dx.doi.org/10.1007/s00521-022-07245-x> (cit. on p. 37).
- [364] M. Afshari Nia, F. Panahi, and M. Ehteram. “Convolutional Neural Network- ANN- E (Tanh): A New Deep Learning Model for Predicting Rainfall.” In: *Water Resources Management* 37.4 (Feb. 2023), pp. 1785–1810. ISSN: 1573-1650. DOI: 10.1007/s11269-023-03454-8. URL: <http://dx.doi.org/10.1007/s11269-023-03454-8> (cit. on p. 37).
- [365] G. Chen, Q. Wang, X. Li, and Y. Zhang. “Target detection based on a new triple activation function.” In: *Systems Science & Control Engineering* 10.1 (June 2022), pp. 629–635. ISSN: 2164-2583. DOI: 10.1080/21642583.2022.2091060. URL: <http://dx.doi.org/10.1080/21642583.2022.2091060> (cit. on p. 37).
- [366] Z. Wang, H. Liu, F. Liu, and D. Gao. “Why KDAC? A general activation function for knowledge discovery.” In: *Neurocomputing* 501 (Aug. 2022), pp. 343–358. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.06.019. URL: <http://dx.doi.org/10.1016/j.neucom.2022.06.019> (cit. on p. 38).
- [367] C. Xiao, P. Zhong, and C. Zheng. “Enhancing Adversarial Defense by k-Winners-Take-All.” In: *International Conference on Learning Representations*. 2020. URL: <https://openreview.net/forum?id=Skgyv64tvr> (cit. on p. 38).
- [368] F. Kayim and A. Yilmaz. “Time Series Forecasting With Volatility Activation Function.” In: *IEEE Access* 10 (2022), pp. 104000–104010. ISSN: 2169-3536. DOI: 10.1109/access.2022.3211312. URL: <http://dx.doi.org/10.1109/ACCESS.2022.3211312> (cit. on p. 38).
- [369] N. Xin, J. Su, and M. M. Hasan. “Multivariate Time Series Spatial Extreme Clustering with Voformer-Ec Neural Networks.” In: (2023). DOI: 10.2139/ssrn.4502409. URL: <http://dx.doi.org/10.2139/ssrn.4502409> (cit. on p. 38).
- [370] S. Reid and K. Ferens. “A Hybrid Chaotic Activation Function for Artificial Neural Networks.” In: *Advances in Artificial Intelligence and Applied Cognitive Computing*. Springer International Publishing, 2021, pp. 1097–1105. ISBN: 9783030702960. DOI: 10.1007/978-3-030-70296-0_87. URL: http://dx.doi.org/10.1007/978-3-030-70296-0_87 (cit. on p. 39).
- [371] A. N. M. E. Kabir, A. F. M. N. uddin, M. Asaduzzaman, M. F. Hasan, M. I. Hasan, and M. Shahjahan. “Fusion of Chaotic Activation Functions in training neural network.” In: *2012 7th International Conference on Electrical and Computer Engineering*. IEEE, Dec. 2012. DOI: 10.1109/icece.2012.6471592. URL: <http://dx.doi.org/10.1109/ICECE.2012.6471592> (cit. on p. 39).
- [372] H. Abbasi, M. Yaghoobi, M. Teshnehlab, and A. Sharifi. “Cascade chaotic neural network (CCNN): a new model.” In: *Neural Computing and Applications* 34.11 (Jan. 2022), pp. 8897–8917. ISSN: 1433-3058. DOI: 10.1007/s00521-022-06912-3. URL: <http://dx.doi.org/10.1007/s00521-022-06912-3> (cit. on p. 39).
- [373] Y. Wu, M. Zhao, and X. Ding. “Beyond weights adaptation: a new neuron model with trainable activation function and its supervised learning.” In: *Proceedings of International Conference on Neural Networks (ICNN’97)*. ICNN-97. IEEE, 1997. DOI: 10.1109/icnn.1997.616194. URL: <http://dx.doi.org/10.1109/ICNN.1997.616194> (cit. on pp. 39, 73).
- [374] Y. Wu and M. Zhao. “A neuron model with trainable activation function (TAF) and its MFNN supervised learning.” In: *Science in China Series F Information Sciences* 44.5 (Oct. 2001), pp. 366–375. ISSN: 1862-2836. DOI: 10.1007/bf02714739. URL: <http://dx.doi.org/10.1007/BF02714739> (cit. on p. 39).
- [375] S. Flennerhag et al. “Breaking the Activation Function Bottleneck through Adaptive Parameterization.” In: *CoRR* abs/1805.08574 (2018). arXiv: 1805.08574. URL: <http://arxiv.org/abs/1805.08574> (cit. on p. 39).
- [376] L. Vecchi et al. “Learning and Approximation Capabilities of Adaptive Spline Activation Function Neural Networks.” In: *Neural Networks* 11.2 (1998), pp. 259–270. DOI: 10.1016/S0893-6080(97)00118-4. URL: [https://doi.org/10.1016/S0893-6080\(97\)00118-4](https://doi.org/10.1016/S0893-6080(97)00118-4) (cit. on pp. 39, 77).
- [377] M. Dushkoff and R. Ptucha. “Adaptive Activation Functions for Deep Networks.” In: *Electronic Imaging* 2016.19 (Feb. 2016), pp. 1–5. DOI: 10.2352/issn.2470-1173.2016.19.coimg-149. URL: <https://doi.org/10.2352/issn.2470-1173.2016.19.coimg-149> (cit. on pp. 39, 74).
- [378] L. Hou et al. “ConvNets with Smooth Adaptive Activation Functions for Regression.” In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. Ed. by A. Singh and J. Zhu. Vol. 54. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 20–22 Apr 2017, pp. 430–439. URL: <http://proceedings.mlr.press/v54/hou17a.html> (cit. on pp. 39, 67).
- [379] S. Scardapane et al. “Learning Activation Functions from Data Using Cubic Spline Interpolation.” In: *Neural Advances in Processing Nonlinear Dynamic Signals*. Springer International Publishing, July 2018, pp. 73–83. DOI: 10.1007/978-3-319-95098-3_7. URL: https://doi.org/10.1007/978-3-319-95098-3_7 (cit. on pp. 39, 77).
- [380] V. Kunc and J. Kléma. “On transformative adaptive activation functions in neural networks for gene expression inference.” In: *PLOS ONE* 16.1 (Jan. 2021). Ed. by H. Fröhlich, e0243915. DOI: 10.1371/journal.pone.0243915. URL: <https://doi.org/10.1371/journal.pone.0243915> (cit. on pp. 40, 85).
- [381] V. Kunc and J. Kléma. “On tower and checkerboard neural network architectures for gene expression inference.” In: *BMC Genomics* 21.S5 (Dec. 2020). DOI: 10.1186/s12864-020-06821-6. URL: <https://doi.org/10.1186/s12864-020-06821-6> (cit. on pp. 40, 85).
- [382] Y.-D. Zhang, C. Pan, J. Sun, and C. Tang. “Multiple sclerosis identification by convolutional neural network with dropout and parametric ReLU.” In: *Journal of Computational Science* 28 (Sept. 2018), pp. 1–10. DOI: 10.1016/j.jocs.2018.07.003. URL: <https://doi.org/10.1016/j.jocs.2018.07.003> (cit. on p. 40).
- [383] F. Leofante, P. Henriksen, and A. Lomuscio. “Verification-friendly Networks: the Case for Parametric ReLUs.” In: *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, June 2023. DOI: 10.1109/ijcnn54540.2023.10191169. URL: <http://dx.doi.org/10.1109/IJCNN54540.2023.10191169> (cit. on p. 40).
- [384] S. Dai, S. Mahloujifar, and P. Mittal. “Parameterizing Activation Functions for Adversarial Robustness.” In: *2022 IEEE Security and Privacy Workshops (SPW)*. IEEE, May 2022. DOI: 10.1109/spw54247.2022.9833884. URL: <http://dx.doi.org/10.1109/SPW54247.2022.9833884> (cit. on pp. 40, 48, 55, 56).
- [385] B. Heo, J. Kim, S. Yun, H. Park, N. Kwak, and J. Y. Choi. “A Comprehensive Overhaul of Feature Distillation.” In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00201. URL: <http://dx.doi.org/10.1109/ICCV.2019.00201> (cit. on p. 40).
- [386] N. Ma, X. Zhang, and J. Sun. “Funnel Activation for Visual Recognition.” In: *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 351–368. ISBN: 9783030586218. DOI: 10.1007/978-3-030-58621-8_21. URL: http://dx.doi.org/10.1007/978-3-030-58621-8_21 (cit. on p. 41).

- [387] J. Gao, J. Yi, and Y. L. Murphey. "Self-Supervised Learning for Driving Maneuver Prediction from Multivariate Temporal Signals." In: *2020 Chinese Automation Congress (CAC)*. IEEE, Nov. 2020. DOI: 10.1109/cac51589.2020.9327088. URL: <http://dx.doi.org/10.1109/CAC51589.2020.9327088> (cit. on p. 41).
- [388] S. Bogoi and A. Udrea. "A Lightweight Deep Learning Approach for Liver Segmentation." In: *Mathematics* 11.1 (Dec. 2022), p. 95. ISSN: 2227-7390. DOI: 10.3390/math11010095. URL: <http://dx.doi.org/10.3390/math11010095> (cit. on p. 41).
- [389] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng. "ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 143–159. ISBN: 9783030585686. DOI: 10.1007/978-3-030-58568-6_9. URL: http://dx.doi.org/10.1007/978-3-030-58568-6_9 (cit. on pp. 41, 59).
- [390] K. Biswas, S. Kumar, S. Banerjee, and A. Kumar Pandey. "SAU: Smooth Activation Function Using Convolution with Approximate Identities." In: *Computer Vision – ECCV 2022*. Springer Nature Switzerland, 2022, pp. 313–329. ISBN: 9783031198038. DOI: 10.1007/978-3-031-19803-8_19. URL: http://dx.doi.org/10.1007/978-3-031-19803-8_19 (cit. on p. 41).
- [391] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. "Smooth Maximum Unit: Smooth Activation Function for Deep Networks using Smoothing Maximum Technique." In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022. DOI: 10.1109/cvpr52688.2022.00087. URL: <http://dx.doi.org/10.1109/CVPR52688.2022.00087> (cit. on pp. 41, 79).
- [392] A. Maniatopoulos and N. Mitianoudis. "Learnable Leaky ReLU (LeLeLU): An Alternative Accuracy-Optimized Activation Function." In: *Information* 12.12 (Dec. 2021), p. 513. ISSN: 2078-2489. DOI: 10.3390/info12120513. URL: <http://dx.doi.org/10.3390/info12120513> (cit. on p. 41).
- [393] D. Kim, J. Kim, and J. Kim. "Elastic exponential linear units for convolutional neural networks." In: *Neurocomputing* 406 (Sept. 2020), pp. 253–266. DOI: 10.1016/j.neucom.2020.03.051. URL: <https://doi.org/10.1016/j.neucom.2020.03.051> (cit. on pp. 42, 53).
- [394] K. Shridhar, J. Lee, H. Hayashi, P. Mehta, B. K. Iwana, S. Kang, S. Uchida, S. Ahmed, and A. Dengel. *ProbAct: A Probabilistic Activation Function for Deep Neural Networks*. 2019. DOI: 10.48550/ARXIV.1905.10761. URL: <https://arxiv.org/abs/1905.10761> (cit. on p. 42).
- [395] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio. "Noisy Activation Functions." In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 3059–3068. URL: <https://proceedings.mlr.press/v48/gulcehre16.html> (cit. on pp. 42, 61, 62, 75).
- [396] S. Reid, K. Ferens, and W. Kinsner. "Adaptive Chaotic Injection to Reduce Overfitting in Artificial Neural Networks." In: *2022 IEEE 21st International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*. IEEE, Dec. 2022. DOI: 10.1109/iccicc57084.2022.10101500. URL: http://dx.doi.org/10.1109/ICCI*CC57084.2022.10101500 (cit. on p. 42).
- [397] Y. Jiang, J. Xie, and D. Zhang. "An Adaptive Offset Activation Function for CNN Image Classification Tasks." In: *Electronics* 11.22 (Nov. 2022), p. 3799. DOI: 10.3390/electronics11223799. URL: <https://doi.org/10.3390/electronics11223799> (cit. on p. 42).
- [398] X. Hu, P. Niu, J. Wang, and X. Zhang. "A Dynamic Rectified Linear Activation Units." In: *IEEE Access* 7 (2019), pp. 180409–180416. DOI: 10.1109/access.2019.2959036. URL: <https://doi.org/10.1109/access.2019.2959036> (cit. on pp. 42, 43, 82).
- [399] J. Si, S. L. Harris, and E. Yfantis. "A Dynamic ReLU on Neural Network." In: *2018 IEEE 13th Dallas Circuits and Systems Conference (DCAS)*. IEEE, Nov. 2018. DOI: 10.1109/dcass.2018.8620116. URL: <https://doi.org/10.1109/dcass.2018.8620116> (cit. on pp. 42, 43, 82).
- [400] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu. *Dynamic ReLU*. 2020. DOI: 10.48550/ARXIV.2003.10027. URL: <https://arxiv.org/abs/2003.10027> (cit. on pp. 42, 43, 82).
- [401] S. Qiu, X. Xu, and B. Cai. "FReLU: Flexible Rectified Linear Units for Improving Convolutional Neural Networks." In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, Aug. 2018. DOI: 10.1109/icpr.2018.8546022. URL: <https://doi.org/10.1109/icpr.2018.8546022> (cit. on p. 43).
- [402] W. Yu, C. Si, P. Zhou, M. Luo, Y. Zhou, J. Feng, S. Yan, and X. Wang. "MetaFormer Baselines for Vision." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.2 (Feb. 2024), pp. 896–912. ISSN: 1939-3539. DOI: 10.1109/tpami.2023.3329173. URL: <http://dx.doi.org/10.1109/TPAMI.2023.3329173> (cit. on p. 43).
- [403] D. Chen, J. Li, and K. Xu. *ARReLU: Attention-based Rectified Linear Unit*. 2020. DOI: 10.48550/ARXIV.2006.13858. URL: <https://arxiv.org/abs/2006.13858> (cit. on p. 44).
- [404] P. Gupta, H. Siebert, M. P. Heinrich, and K. T. Rajamani. "DA-AR-Net: an attentive activation based Deformable auto-encoder for group-wise registration." In: *Medical Imaging 2021: Image Processing*. Ed. by B. A. Landman and I. Išgum. SPIE, Feb. 2021. DOI: 10.1117/12.2582176. URL: <https://doi.org/10.1117/12.2582176> (cit. on p. 44).
- [405] S. Balaji, T. Kavya, and N. Sebastian. "Learn-Able Parameter Guided Activation Functions." In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, Aug. 2020, pp. 583–597. DOI: 10.1007/978-3-030-55180-3_43. URL: https://doi.org/10.1007/978-3-030-55180-3_43 (cit. on pp. 44, 45).
- [406] M. Varshney and P. Singh. "Optimizing nonlinear activation function for convolutional neural networks." In: *Signal, Image and Video Processing* 15.6 (Feb. 2021), pp. 1323–1330. DOI: 10.1007/s11760-021-01863-z. URL: <https://doi.org/10.1007/s11760-021-01863-z> (cit. on pp. 44, 45).
- [407] J. Inturrisi, S. Y. Khoo, A. Kouzani, and R. Pagliarella. *Piecewise Linear Units Improve Deep Neural Networks*. 2021. DOI: 10.48550/ARXIV.2108.00700. URL: <https://arxiv.org/abs/2108.00700> (cit. on p. 44).
- [408] Z. Tang, L. Luo, H. Peng, and S. Li. "A joint residual network with paired ReLUs activation for image super-resolution." In: *Neurocomputing* 273 (Jan. 2018), pp. 37–46. DOI: 10.1016/j.neucom.2017.07.061. URL: <https://doi.org/10.1016/j.neucom.2017.07.061> (cit. on p. 45).
- [409] A. Rozsa and T. E. Boulton. *Improved Adversarial Robustness by Reducing Open Space Risk via Tent Activations*. 2019. DOI: 10.48550/ARXIV.1908.02435. URL: <https://arxiv.org/abs/1908.02435> (cit. on p. 45).
- [410] J. Wang, J. Xu, and J. Zhu. "CNNs with Compact Activation Function." In: *Lecture Notes in Computer Science*. Springer International Publishing, 2022, pp. 319–327. ISBN: 9783031087547. DOI: 10.1007/978-3-031-08754-7_40. URL: http://dx.doi.org/10.1007/978-3-031-08754-7_40 (cit. on pp. 45, 46).
- [411] Q. Hong, J. W. Siegel, Q. Tan, and J. Xu. *On the Activation Function Dependence of the Spectral Bias of Neural Networks*. 2022. DOI: 10.48550/ARXIV.2208.04924. URL: <https://arxiv.org/abs/2208.04924> (cit. on p. 46).
- [412] Y. Yu, K. Adu, N. Tashi, P. Anokye, X. Wang, and M. A. Ayidzoe. "RMAF: Relu-Memristor-Like Activation Function for Deep Learning." In: *IEEE Access* 8 (2020), pp. 72727–72741. DOI: 10.1109/access.2020.2987829. URL: <https://doi.org/10.1109/access.2020.2987829> (cit. on p. 46).
- [413] A. Gupta and R. Duggal. "P-TELU: Parametric Tan Hyperbolic Linear Unit Activation for Deep Neural Networks." In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. IEEE, Oct. 2017. DOI: 10.1109/iccvw.2017.119. URL: <https://doi.org/10.1109/iccvw.2017.119> (cit. on p. 46).
- [414] M. A. Mercioni and S. Holban. "Developing Novel Activation Functions in Time Series Anomaly Detection with LSTM Autoencoder." In: *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, May 2021. DOI: 10.1109/saci51354.2021.9465604. URL: <http://dx.doi.org/10.1109/SACI51354.2021.9465604> (cit. on p. 46).
- [415] S.-L. Shen, N. Zhang, A. Zhou, and Z.-Y. Yin. "Enhancement of neural networks with an alternative activation function tanhLU." In: *Expert Systems with Applications* 199 (Aug. 2022), p. 117181. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.117181. URL: <http://dx.doi.org/10.1016/j.eswa.2022.117181> (cit. on p. 46).
- [416] T. Zhang, J. Yang, W.-a. Song, and C.-f. Song. "Research on Improved Activation Function TRReLU." In: *Journal of Chinese Computer Systems* 40.1, 58 (2019), pp. 58–63. URL: <http://zxwt.sict.ac.cn/EN/Y2019/V40/I1/58> (cit. on p. 47).

- [417] M. Nakhua, D. Bavishi, S. Tikoo, and S. Khedkar. "TReLU: A Novel Activation Function for Modern Day Intrusion Detection System Using Deep Neural Networks." In: *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, July 2023. DOI: 10.1109/icccnt56998.2023.10306887. URL: <http://dx.doi.org/10.1109/icccnt56998.2023.10306887> (cit. on p. 47).
- [418] X. Wang, Y. Qin, Y. Wang, S. Xiang, and H. Chen. "ReLUanh: An activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis." In: *Neurocomputing* 363 (Oct. 2019), pp. 88–98. DOI: 10.1016/j.neucom.2019.07.017. URL: <https://doi.org/10.1016/j.neucom.2019.07.017> (cit. on p. 47).
- [419] L. B. Godfrey. "An Evaluation of Parametric Activation Functions for Deep Learning." In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, Oct. 2019. DOI: 10.1109/smc.2019.8913972. URL: <https://doi.org/10.1109/smc.2019.8913972> (cit. on pp. 47, 48).
- [420] L. Zhang, T. Yang, R. Jin, and X. He. "O(logT) Projections for Stochastic Optimization of Smooth and Strongly Convex Functions." In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1121–1129. URL: <https://proceedings.mlr.press/v28/zhang13e.html> (cit. on p. 48).
- [421] T. Yang. "Trading Computation for Communication: Distributed Stochastic Dual Coordinate Ascent." In: *Advances in Neural Information Processing Systems*. Ed. by C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/dc912a253d1e9ba40e2c597ed2376640-Paper.pdf (cit. on p. 48).
- [422] E. H. Bergou, Y. Diouane, V. Kunc, V. Kungurtsev, and C. W. Royer. "A Subsampling Line-Search Method with Second-Order Results." In: *INFORMS Journal on Optimization* 4.4 (Oct. 2022), pp. 403–425. DOI: 10.1287/ijoo.2022.0072. URL: <https://doi.org/10.1287/ijoo.2022.0072> (cit. on p. 48).
- [423] M. Mahdavi, L. Zhang, and R. Jin. "Mixed Optimization for Smooth Functions." In: *Advances in Neural Information Processing Systems*. Ed. by C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/f73b76ce8949fe29bf2a537cfa420e8f-Paper.pdf (cit. on p. 48).
- [424] S. Hayou, A. Doucet, and J. Rousseau. "On the Impact of the Activation function on Deep Neural Networks Training." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2672–2680. URL: <https://proceedings.mlr.press/v97/hayou19a.html> (cit. on p. 48).
- [425] M. A. Mercioni and S. Holban. "Soft Clipping Mish - A Novel Activation Function for Deep Learning." In: *2021 4th International Conference on Information and Computer Technologies (ICICT)*. IEEE, Mar. 2021. DOI: 10.1109/icict52872.2021.00010. URL: <http://dx.doi.org/10.1109/ICICT52872.2021.00010> (cit. on p. 48).
- [426] M. A. Mercioni and S. Holban. "Prediction of Machine Temperature System Failure Using a Novel Activation Function." In: *2022 International Symposium on Electronics and Telecommunications (ISETC)*. IEEE, Nov. 2022. DOI: 10.1109/isetc56213.2022.10010046. URL: <http://dx.doi.org/10.1109/ISETC56213.2022.10010046> (cit. on p. 48).
- [427] M. A. Mercioni and S. Holban. "Weather Forecasting Modeling Using Soft-Clipping Swish Activation Function." In: *2022 IEEE 16th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, May 2022. DOI: 10.1109/saci55618.2022.9919575. URL: <http://dx.doi.org/10.1109/SACI55618.2022.9919575> (cit. on p. 48).
- [428] M. A. Mercioni and S. Holban. "Soft-Clipping Swish: A Novel Activation Function for Deep Learning." In: *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, May 2021. DOI: 10.1109/saci51354.2021.9465622. URL: <http://dx.doi.org/10.1109/SACI51354.2021.9465622> (cit. on p. 48).
- [429] M. A. Mercioni and S. Holban. "P-Swish: Activation Function with Learnable Parameters Based on Swish Activation Function in Deep Learning." In: *2020 International Symposium on Electronics and Telecommunications (ISETC)*. IEEE, Nov. 2020. DOI: 10.1109/isetc50328.2020.9301059. URL: <http://dx.doi.org/10.1109/ISETC50328.2020.9301059> (cit. on p. 48).
- [430] L. Shi and X. Xie. "Image Segmentation Method for Maize Ear Using Self-defined Activation Function." In: *Procedia Computer Science* 208 (2022), pp. 162–169. ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.10.024. URL: <http://dx.doi.org/10.1016/j.procs.2022.10.024> (cit. on p. 48).
- [431] L. Trotter, P. Giguère, and B. Chaib-draa. *Parametric Exponential Linear Unit for Deep Convolutional Neural Networks*. 2016. DOI: 10.48550/ARXIV.1605.09332. URL: <https://arxiv.org/abs/1605.09332> (cit. on pp. 49, 85).
- [432] S. Qian et al. "Adaptive activation functions in convolutional neural networks." In: *Neurocomputing* 272 (Jan. 2018), pp. 204–212. DOI: 10.1016/j.neucom.2017.06.070. URL: <https://doi.org/10.1016/j.neucom.2017.06.070> (cit. on pp. 49, 64).
- [433] B. Catalbaş and Ö. Morgül. "Deep learning with Extended Exponential Linear Unit (DELU)." In: *Neural Computing and Applications* 35.30 (Aug. 2023), pp. 22705–22724. ISSN: 1433-3058. DOI: 10.1007/s00521-023-08932-z. URL: <http://dx.doi.org/10.1007/s00521-023-08932-z> (cit. on p. 49).
- [434] Z. Qiumei, T. Dan, and W. Fenghua. "Improved Convolutional Neural Network Based on Fast Exponentially Linear Unit Activation Function." In: *IEEE Access* 7 (2019), pp. 151359–151367. DOI: 10.1109/access.2019.2948112. URL: <https://doi.org/10.1109/access.2019.2948112> (cit. on p. 49).
- [435] N. N. Schraudolph. "A Fast, Compact Approximation of the Exponential Function." In: *Neural Computation* 11.4 (May 1999), pp. 853–862. DOI: 10.1162/089976699300016467. URL: <https://doi.org/10.1162/089976699300016467> (cit. on p. 49).
- [436] K. Adem. "P+FEU: Flexible and trainable fast exponential linear unit for deep learning architectures." In: *Neural Computing and Applications* 34.24 (July 2022), pp. 21729–21740. ISSN: 1433-3058. DOI: 10.1007/s00521-022-07625-3. URL: <http://dx.doi.org/10.1007/s00521-022-07625-3> (cit. on p. 50).
- [437] Y. Li, C. Fan, Y. Li, Q. Wu, and Y. Ming. "Improving deep neural network with Multiple Parametric Exponential Linear Units." In: *Neurocomputing* 301 (Aug. 2018), pp. 11–24. DOI: 10.1016/j.neucom.2018.01.084. URL: <https://doi.org/10.1016/j.neucom.2018.01.084> (cit. on pp. 50, 53).
- [438] C.-H. Pham, C. Tor-Díez, H. Meunier, N. Bednarek, R. Fablet, N. Passat, and F. Rousseau. "Multiscale brain MRI super-resolution using deep 3D convolutional networks." In: *Computerized Medical Imaging and Graphics* 77 (Oct. 2019), p. 101647. DOI: 10.1016/j.compmedimag.2019.101647. URL: <https://doi.org/10.1016/j.compmedimag.2019.101647> (cit. on p. 50).
- [439] M. Lin et al. "Network In Network." In: *CoRR* abs/1312.4400 (2013). arXiv: 1312.4400. URL: <http://arxiv.org/abs/1312.4400> (cit. on pp. 50, 81).
- [440] R. Jie, J. Gao, A. Vasnev, and M.-n. Tran. "Regularized Flexible Activation Function Combination for Deep Neural Networks." In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Jan. 2021. DOI: 10.1109/icpr48806.2021.9412370. URL: <http://dx.doi.org/10.1109/ICPR48806.2021.9412370> (cit. on pp. 50, 60).
- [441] L. B. Godfrey and M. S. Gashler. "A continuum among logarithmic, linear, and exponential functions, and its potential to improve generalization in neural networks." In: *2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 2015, pp. 481–486. ISBN: 978-9-8975-8164-9. URL: <https://ieeexplore.ieee.org/document/7526959/> (cit. on p. 50).
- [442] J. T. Barron. *Continuously Differentiable Exponential Linear Units*. 2017. DOI: 10.48550/ARXIV.1704.07483. URL: <https://arxiv.org/abs/1704.07483> (cit. on p. 51).
- [443] A. Rajanand and P. Singh. *ErfReLU: Adaptive Activation Function for Deep Neural Network*. 2023. DOI: 10.48550/ARXIV.2306.01822. URL: <https://arxiv.org/abs/2306.01822> (cit. on p. 51).
- [444] K. Pratama and D.-K. Kang. "Trainable activation function with differentiable negative side and adaptable rectified point." In: *Applied Intelligence* 51.3 (Oct. 2020), pp. 1784–1801. DOI: 10.1007/s10489-020-01885-z. URL: <https://doi.org/10.1007/s10489-020-01885-z> (cit. on p. 51).
- [445] B. Grelsson and M. Felsberg. "Improved Learning in Convolutional Neural Networks with Shifted Exponential Linear Units (ShELUs)." In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, Aug. 2018. DOI: 10.1109/icpr.2018.8545104. URL: <https://doi.org/10.1109/icpr.2018.8545104> (cit. on pp. 51, 52).

- [446] I. Javid, R. Ghazali, I. Syed, N. A. Husaini, and M. Zulqarnain. "Developing Novel T-Swish Activation Function in Deep Learning." In: *2022 International Conference on IT and Industrial Technologies (ICIT)*. IEEE, Oct. 2022. DOI: 10.1109/icit56493.2022.9989151. URL: <http://dx.doi.org/10.1109/ICIT56493.2022.9989151> (cit. on p. 52).
- [447] A. M. Atto, S. Galichet, D. Pastor, and N. Méger. "On joint parameterizations of linear and nonlinear functionals in neural networks." In: *Neural Networks* 160 (Mar. 2023), pp. 12–21. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2022.12.019. URL: <http://dx.doi.org/10.1016/j.neunet.2022.12.019> (cit. on p. 52).
- [448] A. M. Atto, D. Pastor, and G. Mercier. "Smooth sigmoid wavelet shrinkage for non-parametric estimation." In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, Mar. 2008. DOI: 10.1109/icassp.2008.4518347. URL: <http://dx.doi.org/10.1109/ICASSP.2008.4518347> (cit. on p. 52).
- [449] Q. Cheng, H. Li, Q. Wu, L. Ma, and K. N. Ngan. "Parametric Deformable Exponential Linear Units for deep neural networks." In: *Neural Networks* 125 (May 2020), pp. 281–289. DOI: 10.1016/j.neunet.2020.02.012. URL: <https://doi.org/10.1016/j.neunet.2020.02.012> (cit. on pp. 52, 53).
- [450] B. Duan, Y. Yang, and X. Dai. "Feature Activation through First Power Linear Unit with Sign." In: *Electronics* 11.13 (June 2022), p. 1980. ISSN: 2079-9292. DOI: 10.3390/electronics11131980. URL: <http://dx.doi.org/10.3390/electronics11131980> (cit. on p. 53).
- [451] T. Yamada and T. Yabuta. "Neural network controller using autotuning method for nonlinear functions." In: *IEEE Transactions on Neural Networks* 3.4 (July 1992), pp. 595–601. DOI: 10.1109/72.143373. URL: <https://doi.org/10.1109/72.143373> (cit. on pp. 53, 54).
- [452] C.-T. Chen and W.-D. Chang. "A feedforward neural network with function shape autotuning." In: *Neural Networks* 9.4 (June 1996), pp. 627–641. DOI: 10.1016/0893-6080(96)00006-8. URL: [https://doi.org/10.1016/0893-6080\(96\)00006-8](https://doi.org/10.1016/0893-6080(96)00006-8) (cit. on p. 54).
- [453] E. Trentin. "Networks with trainable amplitude of activation functions." In: *Neural Networks* 14.4-5 (May 2001), pp. 471–493. DOI: 10.1016/S0893-6080(01)00028-4. URL: [https://doi.org/10.1016/S0893-6080\(01\)00028-4](https://doi.org/10.1016/S0893-6080(01)00028-4) (cit. on p. 54).
- [454] S. L. Goh and D. P. Mandic. "Recurrent neural networks with trainable amplitude of activation functions." In: *Neural Networks* 16.8 (Oct. 2003), pp. 1095–1100. DOI: 10.1016/S0893-6080(03)00139-4. URL: [https://doi.org/10.1016/S0893-6080\(03\)00139-4](https://doi.org/10.1016/S0893-6080(03)00139-4) (cit. on p. 54).
- [455] T. Yamada and T. Yabuta. "Remarks on a neural network controller which uses an auto-tuning method for nonlinear functions." In: *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*. IEEE, 1992. DOI: 10.1109/ijcnn.1992.226893. URL: <http://dx.doi.org/10.1109/IJCNN.1992.226893> (cit. on p. 54).
- [456] N. M. Nawi, R. Ransing, M. N. M. Salleh, R. Ghazali, and N. A. Hamid. "The effect of gain variation in improving learning speed of back propagation neural network algorithm on classification problems." In: *Symposium on Progress in Information & Communication Technology*. 2009 (cit. on p. 54).
- [457] S. K. Sharma and P. Chandra. "An Adaptive Sigmoidal Activation Function Cascading Neural Networks." In: *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011*. Springer Berlin Heidelberg, 2011, pp. 105–116. ISBN: 9783642196447. DOI: 10.1007/978-3-642-19644-7_12. URL: http://dx.doi.org/10.1007/978-3-642-19644-7_12 (cit. on p. 54).
- [458] M. A. Mercioni, A. Tiron, and S. Holban. "Dynamic Modification of Activation Function using the Backpropagation Algorithm in the Artificial Neural Networks." In: *International Journal of Advanced Computer Science and Applications* 10.4 (2019). ISSN: 2158-107X. DOI: 10.14569/ijacsa.2019.0100406. URL: <http://dx.doi.org/10.14569/ijacsa.2019.0100406> (cit. on p. 54).
- [459] Y. Bai et al. "The performance of the backpropagation algorithm with varying slope of the activation function." In: *Chaos, Solitons & Fractals* 40.1 (Apr. 2009), pp. 69–77. DOI: 10.1016/j.chaos.2007.07.033. URL: <https://doi.org/10.1016/j.chaos.2007.07.033> (cit. on p. 54).
- [460] C.-C. Yu et al. "An adaptive activation function for multilayer feedforward neural networks." In: *2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM '02. Proceedings*. IEEE, 2002. DOI: 10.1109/tencon.2002.1181357. URL: <https://doi.org/10.1109/tencon.2002.1181357> (cit. on p. 54).
- [461] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. *TanhSoft – a family of activation functions combining Tanh and Softplus*. 2020. DOI: 10.48550/ARXIV.2009.03863. URL: <https://arxiv.org/abs/2009.03863> (cit. on p. 54).
- [462] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. "TanhSoft—Dynamic Trainable Activation Functions for Faster Learning and Better Performance." In: *IEEE Access* 9 (2021), pp. 120613–120623. ISSN: 2169-3536. DOI: 10.1109/access.2021.3105355. URL: <http://dx.doi.org/10.1109/ACCESS.2021.3105355> (cit. on p. 54).
- [463] Y. Ying, N. Zhang, P. Shan, L. Miao, P. Sun, and S. Peng. "PSigmoid: Improving squeeze-and-excitation block with parametric sigmoid." In: *Applied Intelligence* 51.10 (Mar. 2021), pp. 7427–7439. ISSN: 1573-7497. DOI: 10.1007/s10489-021-02247-z. URL: <http://dx.doi.org/10.1007/s10489-021-02247-z> (cit. on p. 55).
- [464] Y. Özbay and G. Tezel. "A new method for classification of ECG arrhythmias using neural network with adaptive activation function." In: *Digital Signal Processing* 20.4 (July 2010), pp. 1040–1049. ISSN: 1051-2004. DOI: 10.1016/j.dsp.2009.10.016. URL: <http://dx.doi.org/10.1016/j.dsp.2009.10.016> (cit. on pp. 55, 71).
- [465] P. Chandra and Y. Singh. "An activation function adapting training algorithm for sigmoidal feedforward networks." In: *Neurocomputing* 61 (Oct. 2004), pp. 429–437. DOI: 10.1016/j.neucom.2004.04.001. URL: <https://doi.org/10.1016/j.neucom.2004.04.001> (cit. on p. 55).
- [466] Y. Singh and P. Chandra. "A class +1 sigmoidal activation functions for FFANNs." In: *Journal of Economic Dynamics and Control* 28.1 (Oct. 2003), pp. 183–187. ISSN: 0165-1889. DOI: 10.1016/S0165-1889(02)00157-4. URL: [http://dx.doi.org/10.1016/S0165-1889\(02\)00157-4](http://dx.doi.org/10.1016/S0165-1889(02)00157-4) (cit. on p. 55).
- [467] P. Chandra and Y. Singh. "A case for the self-adaptation of activation functions in FFANNs." In: *Neurocomputing* 56 (Jan. 2004), pp. 447–454. DOI: 10.1016/j.neucom.2003.08.005. URL: <https://doi.org/10.1016/j.neucom.2003.08.005> (cit. on p. 55).
- [468] P. Chandra. "Sigmoidal Function Classes for Feedforward Artificial Neural Networks." In: *Neural Processing Letters* 18.3 (Dec. 2003), pp. 205–215. DOI: 10.1023/b:nep1.0000011137.04221.96. URL: <https://doi.org/10.1023/b:nep1.0000011137.04221.96> (cit. on p. 55).
- [469] T. Zhang, S. Liu, Y. Wei, and H. Zhang. "A novel feature adaptive extraction method based on deep learning for bearing fault diagnosis." In: *Measurement* 185 (Nov. 2021), p. 110030. ISSN: 0263-2241. DOI: 10.1016/j.measurement.2021.110030. URL: <http://dx.doi.org/10.1016/j.measurement.2021.110030> (cit. on p. 55).
- [470] N. E. Protonotarios, A. S. Fokas, G. A. Kastis, and N. Dikaos. "Sigmoid and Beyond: Algebraic Activation Functions for Artificial Neural Networks Based on Solutions of a Riccati Equation." In: *IT Professional* 24.5 (Sept. 2022), pp. 30–36. ISSN: 1941-045X. DOI: 10.1109/mitp.2022.3204904. URL: <http://dx.doi.org/10.1109/mitp.2022.3204904> (cit. on p. 55).
- [471] N. Ma, X. Zhang, M. Liu, and J. Sun. "Activate or Not: Learning Customized Activation." In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2021. DOI: 10.1109/cvpr46437.2021.00794. URL: <http://dx.doi.org/10.1109/CVPR46437.2021.00794> (cit. on pp. 55, 56).
- [472] Y. Bodyanskiy and S. Kostiuik. "Adaptive hybrid activation function for deep neural networks." In: *System research and information technologies* 1 (Apr. 2022), pp. 87–96. ISSN: 1681-6048. DOI: 10.20535/srit.2308-8893.2022.1.07. URL: <http://dx.doi.org/10.20535/SRIT.2308-8893.2022.1.07> (cit. on p. 56).
- [473] E. Alcaide. *E-swish: Adjusting Activations to Different Network Depths*. 2018. DOI: 10.48550/ARXIV.1801.07145. URL: <https://arxiv.org/abs/1801.07145> (cit. on p. 56).
- [474] S. Zagoruyko and N. Komodakis. "Wide Residual Networks." In: *Proceedings of the British Machine Vision Conference 2016*. British Machine Vision Association, 2016. DOI: 10.5244/c.30.87. URL: <https://doi.org/10.5244/c.30.87> (cit. on pp. 56, 68).

- [475] R. Zhang, K. Zheng, P. Shi, Y. Mei, H. Li, and T. Qiu. "Traffic Sign Detection Based on the Improved YOLOv5." In: *Applied Sciences* 13.17 (Aug. 2023), p. 9748. ISSN: 2076-3417. DOI: 10.3390/app13179748. URL: <http://dx.doi.org/10.3390/app13179748> (cit. on p. 56).
- [476] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors." In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2023. DOI: 10.1109/cvpr52729.2023.00721. URL: <http://dx.doi.org/10.1109/CVPR52729.2023.00721> (cit. on p. 56).
- [477] Y. Ye, Q. Liu, L. Li, Z. Zhang, L. Xu, J. Chu, and B. Wen. "Improving insulator fault detection with effective-YOLOv7 network." In: *Journal of Electronic Imaging* 32.06 (Nov. 2023). ISSN: 1017-9909. DOI: 10.1117/1.jei.32.6.063021. URL: <http://dx.doi.org/10.1117/1.jei.32.6.063021> (cit. on p. 56).
- [478] S. Kan, W. Fang, J. Wu, and V. S. Sheng. "Real-Time Domestic Garbage Detection Method Based on Improved YOLOv5." In: *Communications in Computer and Information Science*. Springer International Publishing, 2022, pp. 62–74. ISBN: 9783031067679. DOI: 10.1007/978-3-031-06767-9_5. URL: http://dx.doi.org/10.1007/978-3-031-06767-9_5 (cit. on p. 56).
- [479] G. Tu, J. Qin, and N. Xiong. "Algorithm of Computer Mainboard Quality Detection for Real-Time Based on QD-YOLO." In: *Electronics* 11.15 (Aug. 2022), p. 2424. ISSN: 2079-9292. DOI: 10.3390/electronics11152424. URL: <http://dx.doi.org/10.3390/electronics11152424> (cit. on p. 56).
- [480] X. Xi, Y. Wu, C. Xia, and S. He. "Feature fusion for object detection at one map." In: *Image and Vision Computing* 123 (July 2022), p. 104466. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2022.104466. URL: <http://dx.doi.org/10.1016/j.imavis.2022.104466> (cit. on p. 56).
- [481] H. Li, L. Wang, and S. Cheng. "HARNU-Net: Hierarchical Attention Residual Nested U-Net for Change Detection in Remote Sensing Images." In: *Sensors* 22.12 (June 2022), p. 4626. ISSN: 1424-8220. DOI: 10.3390/s22124626. URL: <http://dx.doi.org/10.3390/s22124626> (cit. on p. 56).
- [482] J. Wu, J. Li, R. Li, X. Xi, D. Gui, and J. Yin. "A Fast Maritime Target Identification Algorithm for Offshore Ship Detection." In: *Applied Sciences* 12.10 (May 2022), p. 4938. ISSN: 2076-3417. DOI: 10.3390/app12104938. URL: <http://dx.doi.org/10.3390/app12104938> (cit. on p. 56).
- [483] Q. Niu, Y. Wang, S. Yuan, K. Li, and X. Wang. "An Indoor Pool Drowning Risk Detection Method Based on Improved YOLOv4." In: *2022 IEEE 5th Advanced Information Management, Communication, Electronic and Automation Control Conference (IMCEC)*. IEEE, Dec. 2022. DOI: 10.1109/imcec55388.2022.10020040. URL: <http://dx.doi.org/10.1109/IMCEC55388.2022.10020040> (cit. on p. 56).
- [484] B. Zhang, W. Chen, X. Wang, and C. Zhao. "A Lightweight Detection Method of Smartphone Assembly Parts." In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 330–342. ISBN: 9783031204975. DOI: 10.1007/978-3-031-20497-5_27. URL: http://dx.doi.org/10.1007/978-3-031-20497-5_27 (cit. on p. 56).
- [485] K.-Y. Cao, X. Cui, and J.-C. Piao. "Smaller Target Detection Algorithms Based on YOLOv5 in Safety Helmet Wearing Detection." In: *2022 4th International Conference on Robotics and Computer Vision (ICRCV)*. IEEE, Sept. 2022. DOI: 10.1109/icrcv55858.2022.9953233. URL: <http://dx.doi.org/10.1109/ICRCV55858.2022.9953233> (cit. on p. 56).
- [486] Y. Jia, J. Zhao, and L. Yu. "AADH-YOLOv5: improved YOLOv5 based on adaptive activate decoupled head for garbage detection." In: *Journal of Electronic Imaging* 32.04 (July 2023). ISSN: 1017-9909. DOI: 10.1117/1.jei.32.4.043017. URL: <http://dx.doi.org/10.1117/1.jei.32.4.043017> (cit. on p. 56).
- [487] H. Xu, W. Zheng, F. Liu, P. Li, and R. Wang. "Unmanned Aerial Vehicle Perspective Small Target Recognition Algorithm Based on Improved YOLOv5." In: *Remote Sensing* 15.14 (July 2023), p. 3583. ISSN: 2072-4292. DOI: 10.3390/rs15143583. URL: <http://dx.doi.org/10.3390/rs15143583> (cit. on p. 56).
- [488] J. He, J. Duan, Z. Yang, J. Ou, X. Ou, S. Yu, M. Xie, Y. Luo, H. Wang, and Q. Jiang. "Method for Segmentation of Banana Crown Based on Improved DeepLabv3+." In: *Agronomy* 13.7 (July 2023), p. 1838. ISSN: 2073-4395. DOI: 10.3390/agronomy13071838. URL: <http://dx.doi.org/10.3390/agronomy13071838> (cit. on p. 56).
- [489] H. Qin, J. Pan, J. Li, and F. Huang. "Fault Diagnosis Method of Rolling Bearing Based on CBAM_ResNet and ACON Activation Function." In: *Applied Sciences* 13.13 (June 2023), p. 7593. ISSN: 2076-3417. DOI: 10.3390/app13137593. URL: <http://dx.doi.org/10.3390/app13137593> (cit. on p. 56).
- [490] N. Chen, Y. Li, Z. Yang, Z. Lu, S. Wang, and J. Wang. "LODNU: lightweight object detection network in UAV vision." In: *The Journal of Supercomputing* 79.9 (Feb. 2023), pp. 10117–10138. ISSN: 1573-0484. DOI: 10.1007/s11227-023-05065-x. URL: <http://dx.doi.org/10.1007/s11227-023-05065-x> (cit. on p. 56).
- [491] Y. Zhao, L. Lu, W. Yang, Q. Li, and X. Zhang. "Lightweight Tennis Ball Detection Algorithm Based on Robomaster EP." In: *Applied Sciences* 13.6 (Mar. 2023), p. 3461. ISSN: 2076-3417. DOI: 10.3390/app13063461. URL: <http://dx.doi.org/10.3390/app13063461> (cit. on p. 56).
- [492] J. Liu, X. Wang, S. Wu, L. Wan, and F. Xie. "Wind turbine fault detection based on deep residual networks." In: *Expert Systems with Applications* 213 (Mar. 2023), p. 119102. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.119102. URL: <http://dx.doi.org/10.1016/j.eswa.2022.119102> (cit. on p. 56).
- [493] Z. Li, X. Yang, K. Shen, F. Jiang, J. Jiang, H. Ren, and Y. Li. "PSGU: Parametric self-circulation gating unit for deep neural networks." In: *Journal of Visual Communication and Image Representation* 80 (Oct. 2021), p. 103294. ISSN: 1047-3203. DOI: 10.1016/j.jvcir.2021.103294. URL: <http://dx.doi.org/10.1016/j.jvcir.2021.103294> (cit. on pp. 56, 57).
- [494] B. Zheng and Z. Wang. "PATS: A New Neural Network Activation Function with Parameter." In: *2020 5th International Conference on Computer and Communication Systems (ICCCS)*. IEEE, May 2020. DOI: 10.1109/icccs49078.2020.9118471. URL: <http://dx.doi.org/10.1109/ICCCS49078.2020.9118471> (cit. on p. 57).
- [495] A. Paul, R. Bandyopadhyay, J. H. Yoon, Z. W. Geem, and R. Sarkar. "SinLU: Sinu-Sigmoidal Linear Unit." In: *Mathematics* 10.3 (Jan. 2022), p. 337. ISSN: 2227-7390. DOI: 10.3390/math10030337. URL: <http://dx.doi.org/10.3390/math10030337> (cit. on p. 57).
- [496] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. "ErfAct and Pserf: Non-monotonic Smooth Trainable Activation Functions." In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.6 (June 2022), pp. 6097–6105. ISSN: 2159-5399. DOI: 10.1609/aaai.v36i6.20557. URL: <http://dx.doi.org/10.1609/aaai.v36i6.20557> (cit. on pp. 57, 58).
- [497] M. Abdool and T. Dear. *Swim: A General-Purpose, High-Performing, and Efficient Activation Function for Locomotion Control Tasks*. 2023. DOI: 10.48550/ARXIV.2303.02640. URL: <https://arxiv.org/abs/2303.02640> (cit. on p. 58).
- [498] V. Terziyan, D. Malyk, M. Golovianko, and V. Branytskyi. "Hyper-flexible Convolutional Neural Networks based on Generalized Lehmer and Power Means." In: *Neural Networks* 155 (Nov. 2022), pp. 177–203. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2022.08.017. URL: <http://dx.doi.org/10.1016/j.neunet.2022.08.017> (cit. on pp. 58, 65).
- [499] Q. Jiang, L. Zhu, C. Shu, and V. Sekar. "Multilayer perceptron neural network activated by adaptive Gaussian radial basis function and its application to predict lid-driven cavity flow." In: *Acta Mechanica Sinica* 37.12 (Dec. 2021), pp. 1757–1772. ISSN: 1614-3116. DOI: 10.1007/s10409-021-01144-5. URL: <http://dx.doi.org/10.1007/s10409-021-01144-5> (cit. on p. 59).
- [500] F. Duan, F. Chapeau-Blondeau, and D. Abbott. "Optimized injection of noise in activation functions to improve generalization of neural networks." In: *Chaos, Solitons & Fractals* 178 (Jan. 2024), p. 114363. ISSN: 0960-0779. DOI: 10.1016/j.chaos.2023.114363. URL: <http://dx.doi.org/10.1016/j.chaos.2023.114363> (cit. on p. 59).
- [501] H. H. Chieng, N. Wahid, and P. Ong. "Parametric Flatten-T swish: an adaptive nonlinear activation function for deep learning." In: *Journal of Information and Communication Technology* 20 (2020). DOI: 10.32890/jict.20.1.2021.9267. URL: <https://doi.org/10.32890/jict.20.1.2021.9267> (cit. on p. 59).
- [502] A. Mondal and V. K. Shrivastava. "A novel Parametric Flatten-p Mish activation function based deep CNN model for brain tumor classification." In: *Computers in Biology and Medicine* 150 (Nov. 2022), p. 106183. ISSN: 0010-4825. DOI: 10.1016/j.combiomed.2022.106183. URL: <http://dx.doi.org/10.1016/j.combiomed.2022.106183> (cit. on p. 59).

- [503] B. Khagi and G.-R. Kwon. “A novel scaled-gamma-tanh (SGT) activation function in 3D CNN applied for MRI classification.” In: *Scientific Reports* 12.1 (Sept. 2022). ISSN: 2045-2322. DOI: 10.1038/s41598-022-19020-y. URL: <http://dx.doi.org/10.1038/s41598-022-19020-y> (cit. on p. 59).
- [504] R. Ding, H. Liu, and X. Zhou. “IE-Net: Information-Enhanced Binary Neural Networks for Accurate Classification.” In: *Electronics* 11.6 (Mar. 2022), p. 937. ISSN: 2079-9292. DOI: 10.3390/electronics11060937. URL: <http://dx.doi.org/10.3390/electronics11060937> (cit. on p. 59).
- [505] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis. “Adaptive activation functions accelerate convergence in deep and physics-informed neural networks.” In: *Journal of Computational Physics* 404 (Mar. 2020), p. 109136. DOI: 10.1016/j.jcp.2019.109136. URL: <https://doi.org/10.1016/j.jcp.2019.109136> (cit. on pp. 60, 85).
- [506] A. D. Jagtap, K. Kawaguchi, and G. E. Karniadakis. “Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks.” In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 476.2239 (July 2020), p. 20200334. DOI: 10.1098/rspa.2020.0334. URL: <https://doi.org/10.1098/rspa.2020.0334> (cit. on p. 60).
- [507] D. S. Kapoor and A. K. Kohli. “Adaptive-Slope Squashing-Function-Based ANN for CSI Estimation and Symbol Detection in SFBC-OFDM System.” In: *Arabian Journal for Science and Engineering* 46.10 (Jan. 2021), pp. 9451–9464. DOI: 10.1007/s13369-020-05207-w. URL: <https://doi.org/10.1007/s13369-020-05207-w> (cit. on p. 60).
- [508] S. S. Husain, E.-J. Ong, and M. Bober. “ACTNET: End-to-End Learning of Feature Activations and Multi-stream Aggregation for Effective Instance Image Retrieval.” In: *International Journal of Computer Vision* 129.5 (Feb. 2021), pp. 1432–1450. ISSN: 1573-1405. DOI: 10.1007/s11263-021-01444-0. URL: <http://dx.doi.org/10.1007/s11263-021-01444-0> (cit. on pp. 60, 61, 70).
- [509] X.-M. Zhou, L.-F. Li, X.-Z. Zheng, and M. Luo. *LAU: A novel two-parameter learnable Logmoid Activation Unit*. 2023. URL: <https://openreview.net/forum?id=uwBUzlm0GS> (cit. on p. 61).
- [510] X.-M. Zhou, L.-F. Li, X.-Z. Zheng, and M. Luo. *A two-parameter learnable Logmoid Activation Unit*. 2023. URL: <https://openreview.net/forum?id=LcXWYmA8Ek> (cit. on p. 61).
- [511] F. Farhadi, V. P. Nia, and A. Lodi. *Activation Adaptation in Neural Networks*. 2019. DOI: 10.48550/ARXIV.1901.09849. URL: <https://arxiv.org/abs/1901.09849> (cit. on p. 61).
- [512] Y. Zhou, D. Li, S. Huo, and S.-Y. Kung. “Shape autotuning activation function.” In: *Expert Systems with Applications* 171 (June 2021), p. 114534. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2020.114534. URL: <http://dx.doi.org/10.1016/j.eswa.2020.114534> (cit. on p. 61).
- [513] J. Z. Zamora-Esquivel, J. A. Cruz Vargas, and P. Lopez-Meyer. “Fractional Adaptation of Activation Functions In Neural Networks.” In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Jan. 2021. DOI: 10.1109/icpr48806.2021.9413338. URL: <http://dx.doi.org/10.1109/ICPR48806.2021.9413338> (cit. on pp. 62, 63).
- [514] J. Zamora-Esquivel, A. D. Rhodes, and L. Nachman. “Fractional Adaptive Linear Units.” In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.8 (June 2022), pp. 8988–8996. ISSN: 2159-5399. DOI: 10.1609/aaai.v36i8.20882. URL: <http://dx.doi.org/10.1609/aaai.v36i8.20882> (cit. on pp. 62, 63).
- [515] M. S. Job, P. H. Bhateja, M. Gupta, K. Bingi, and B. R. Prusty. “Fractional Rectified Linear Unit Activation Function and Its Variants.” In: *Mathematical Problems in Engineering* 2022 (June 2022). Ed. by X. Li, pp. 1–15. ISSN: 1024-123X. DOI: 10.1155/2022/1860779. URL: <http://dx.doi.org/10.1155/2022/1860779> (cit. on pp. 62–64).
- [516] B. Ramadevi, V. R. Kasi, and K. Bingi. “Fractional ordering of activation functions for neural networks: A case study on Texas wind turbine.” In: *Engineering Applications of Artificial Intelligence* 127 (Jan. 2024), p. 107308. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2023.107308. URL: <http://dx.doi.org/10.1016/j.engappai.2023.107308> (cit. on p. 62).
- [517] J. Zamora-Esquivel, A. Cruz Vargas, R. Camacho Perez, P. Lopez Meyer, H. Cordourier, and O. Tickoo. “Adaptive Activation Functions Using Fractional Calculus.” In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, Oct. 2019. DOI: 10.1109/iccvw.2019.00250. URL: <http://dx.doi.org/10.1109/ICCVW.2019.00250> (cit. on p. 62).
- [518] M. D. Ortigueira. *Fractional Calculus for Scientists and Engineers*. Springer Netherlands, 2011. ISBN: 9789400707474. DOI: 10.1007/978-94-007-0747-4. URL: <http://dx.doi.org/10.1007/978-94-007-0747-4> (cit. on p. 62).
- [519] Y. Bodyanskiy and S. Kostjuk. “Learnable Extended Activation Function for Deep Neural Networks.” In: *International Journal of Computing* (Oct. 2023), pp. 311–318. ISSN: 1727-6209. DOI: 10.47839/ijc.22.3.3225. URL: <http://dx.doi.org/10.47839/ijc.22.3.3225> (cit. on p. 65).
- [520] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang. “Extreme Learning Machine for Regression and Multiclass Classification.” In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42.2 (Apr. 2012), pp. 513–529. ISSN: 1941-0492. DOI: 10.1109/tsmcb.2011.2168604. URL: <http://dx.doi.org/10.1109/TSMCB.2011.2168604> (cit. on p. 65).
- [521] R. Siouda, M. Nemissi, and H. Seridi. “Diverse activation functions based-hybrid RBF-ELM neural network for medical classification.” In: *Evolutionary Intelligence* (July 2022). ISSN: 1864-5917. DOI: 10.1007/s12065-022-00758-3. URL: <http://dx.doi.org/10.1007/s12065-022-00758-3> (cit. on p. 65).
- [522] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. “EIS - Efficient and Trainable Activation Functions for Better Accuracy and Performance.” In: *Artificial Neural Networks and Machine Learning – ICANN 2021*. Springer International Publishing, 2021, pp. 260–272. ISBN: 9783030863401. DOI: 10.1007/978-3-030-86340-1_21. URL: http://dx.doi.org/10.1007/978-3-030-86340-1_21 (cit. on p. 66).
- [523] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey. *EIS – a family of activation functions combining Exponential, ISRU, and Sofplus*. 2020. DOI: 10.48550/ARXIV.2009.13501. URL: <https://arxiv.org/abs/2009.13501> (cit. on p. 66).
- [524] T. A. E. Ferreira, M. Mattheakis, and P. Protopapas. *A New Artificial Neuron Proposal with Trainable Simultaneous Local and Global Activation Function*. 2021. DOI: 10.48550/ARXIV.2101.06100. URL: <https://arxiv.org/abs/2101.06100> (cit. on p. 66).
- [525] J. H. De Medeiros Delgado and T. A. E. Ferreira. “Autoencoder performance analysis with adaptive and trainable activation function to compress images.” In: *2022 IEEE Latin American Conference on Computational Intelligence (LA-COI)*. IEEE, Nov. 2022. DOI: 10.1109/la-cci54402.2022.9981644. URL: <http://dx.doi.org/10.1109/LA-CCI54402.2022.9981644> (cit. on p. 66).
- [526] S. Xu and M. Zhang. “Justification of a neuron-adaptive activation function.” In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. IEEE, 2000. DOI: 10.1109/ijcnn.2000.861351. URL: <https://doi.org/10.1109/ijcnn.2000.861351> (cit. on p. 66).
- [527] S. Xu and M. Zhang. “Data Mining — An Adaptive Neural Network Model for Financial Analysis.” In: *Third International Conference on Information Technology and Applications (ICITA '05)*. IEEE, 2005. DOI: 10.1109/icita.2005.109. URL: <https://doi.org/10.1109/icita.2005.109> (cit. on p. 66).
- [528] S. Xu and M. Zhang. “A New Adaptive Neural Network Model for Financial Data Mining.” In: *Advances in Neural Networks – ISNN 2007*. Springer Berlin Heidelberg, 2007, pp. 1265–1273. DOI: 10.1007/978-3-540-72383-7_147. URL: https://doi.org/10.1007/978-3-540-72383-7_147 (cit. on p. 66).
- [529] G. Tezel and Y. Özbay. “A New Neural Network with Adaptive Activation Function for Classification of ECG Arrhythmias.” In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007, pp. 1–8. DOI: 10.1007/978-3-540-74819-9_1. URL: https://doi.org/10.1007/978-3-540-74819-9_1 (cit. on p. 67).
- [530] S. Xu and M. Zhang. “An Adaptive Activation Function for Higher Order Neural Networks.” In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2002, pp. 356–362. DOI: 10.1007/3-540-36187-1_31. URL: https://doi.org/10.1007/3-540-36187-1_31 (cit. on p. 67).

- [531] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi. *Learning Activation Functions to Improve Deep Neural Networks*. 2014. DOI: 10.48550/ARXIV.1412.6830. URL: <https://arxiv.org/abs/1412.6830> (cit. on p. 67).
- [532] S. Aziznejad, H. Gupta, J. Campos, and M. Unser. “Deep Neural Networks With Trainable Activations and Controlled Lipschitz Constant.” In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 4688–4699. DOI: 10.1109/tsp.2020.3014611. URL: <https://doi.org/10.1109/tsp.2020.3014611> (cit. on p. 67).
- [533] M. Tavakoli, F. Agostinelli, and P. Baldi. “SPASH: Learnable activation functions for improving accuracy and adversarial robustness.” In: *Neural Networks* 140 (Aug. 2021), pp. 1–12. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2021.02.023. URL: <http://dx.doi.org/10.1016/j.neunet.2021.02.023> (cit. on p. 67).
- [534] H. Li, W. Ouyang, and X. Wang. “Multi-Bias Non-linear Activation in Deep Neural Networks.” In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 221–229. URL: <https://proceedings.mlr.press/v48/lia16.html> (cit. on pp. 67, 68).
- [535] G. Maguolo, L. Nanni, and S. Ghidoni. “Ensemble of convolutional neural networks trained with different activation functions.” In: *Expert Systems with Applications* 166 (Mar. 2021), p. 114048. DOI: 10.1016/j.eswa.2020.114048. URL: <https://doi.org/10.1016/j.eswa.2020.114048> (cit. on p. 68).
- [536] M. Fakhfakh and L. Chaari. *Bayesian optimization for sparse neural networks with trainable activation functions*. 2023. DOI: 10.48550/ARXIV.2304.04455. URL: <https://arxiv.org/abs/2304.04455> (cit. on p. 68).
- [537] L. Nanni, A. Lumini, S. Ghidoni, and G. Maguolo. “Stochastic Selection of Activation Layers for Convolutional Neural Networks.” In: *Sensors* 20.6 (Mar. 2020), p. 1626. ISSN: 1424-8220. DOI: 10.3390/s20061626. URL: <http://dx.doi.org/10.3390/s20061626> (cit. on p. 68).
- [538] B. Prach and C. H. Lampert. *1-Lipschitz Neural Networks are more expressive with N-Activations*. 2023. DOI: 10.48550/ARXIV.2311.06103. URL: <https://arxiv.org/abs/2311.06103> (cit. on p. 69).
- [539] V. S. Bawa and V. Kumar. “Linearized sigmoidal activation: A novel activation function with tractable non-linear characteristics to boost representation capability.” In: *Expert Systems with Applications* 120 (Apr. 2019), pp. 346–356. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2018.11.042. URL: <http://dx.doi.org/10.1016/j.eswa.2018.11.042> (cit. on p. 69).
- [540] S. Curci, D. C. Mocanu, and M. Pechenizkiy. *Truly Sparse Neural Networks at Scale*. 2021. DOI: 10.48550/ARXIV.2102.01732. URL: <https://arxiv.org/abs/2102.01732> (cit. on p. 69).
- [541] A. Nicolae. *PLU: The Piecewise Linear Unit Activation Function*. 2018. DOI: 10.48550/ARXIV.1809.09534. URL: <https://arxiv.org/abs/1809.09534> (cit. on p. 69).
- [542] R. Mo, K. Xu, L. Liu, L. Liu, and D. Wang. “Adaptive Linear Unit for Accurate Binary Neural Networks.” In: *2022 16th IEEE International Conference on Signal Processing (ICSP)*. IEEE, Oct. 2022. DOI: 10.1109/icsp56322.2022.9965306. URL: <http://dx.doi.org/10.1109/ICSP56322.2022.9965306> (cit. on p. 70).
- [543] T. Mao, Z. Shi, and D.-X. Zhou. “Approximating functions with multi-features by deep convolutional neural networks.” In: *Analysis and Applications* 21.01 (Nov. 2022), pp. 93–125. ISSN: 1793-6861. DOI: 10.1142/s0219530522400085. URL: <http://dx.doi.org/10.1142/S0219530522400085> (cit. on p. 70).
- [544] N. Patwardhan, M. Ingahlakar, and R. Walambe. *ARiA: Utilizing Richard’s Curve for Controlling the Non-monotonicity of the Activation Function in Deep Neural Nets*. 2018. DOI: 10.48550/ARXIV.1805.08878. URL: <https://arxiv.org/abs/1805.08878> (cit. on p. 70).
- [545] F. J. Richards. “A Flexible Growth Function for Empirical Use.” In: *Journal of Experimental Botany* 10.2 (1959), pp. 290–301. DOI: 10.1093/jxb/10.2.290. URL: <https://doi.org/10.1093/jxb/10.2.290> (cit. on p. 70).
- [546] S. Sarkar, S. Agrawal, T. Baker, P. K. R. Maddikunta, and T. R. Gadekallu. “Catalysis of neural activation functions: Adaptive feed-forward training for big data applications.” In: *Applied Intelligence* 52.12 (Mar. 2022), pp. 13364–13383. ISSN: 1573-7497. DOI: 10.1007/s10489-021-03082-y. URL: <http://dx.doi.org/10.1007/s10489-021-03082-y> (cit. on p. 71).
- [547] S. Gu, W. Li, L. V. Gool, and R. Timofte. “Fast Image Restoration With Multi-Bin Trainable Linear Units.” In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00429. URL: <https://doi.org/10.1109/iccv.2019.00429> (cit. on p. 71).
- [548] X. Gao, Y. Li, W. Li, L. Duan, L. Van Gool, L. Benini, and M. Magno. “Learning continuous piecewise non-linear activation functions for deep neural networks.” In: *2023 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, July 2023. DOI: 10.1109/icme55011.2023.00315. URL: <http://dx.doi.org/10.1109/ICME55011.2023.00315> (cit. on pp. 71, 72).
- [549] Y. Zhou, Z. Zhu, and Z. Zhong. “Learning specialized activation functions with the Piecewise Linear Unit.” In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021. DOI: 10.1109/iccv48922.2021.01188. URL: <http://dx.doi.org/10.1109/ICCV48922.2021.01188> (cit. on p. 71).
- [550] Z. Zhu, Y. Zhou, Y. Dong, and Z. Zhong. “PWL: Learning Specialized Activation Functions with the Piecewise Linear Unit.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023), pp. 1–19. ISSN: 1939-3539. DOI: 10.1109/tpami.2023.3286109. URL: <http://dx.doi.org/10.1109/TPAMI.2023.3286109> (cit. on p. 71).
- [551] Z. Zhu and Y. Dong. “Non-uniform Piecewise Linear Activation Functions in Deep Neural Networks.” In: *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, Aug. 2022. DOI: 10.1109/icpr56361.2022.9956345. URL: <http://dx.doi.org/10.1109/ICPR56361.2022.9956345> (cit. on p. 71).
- [552] J. Zhang, S. Zhu, N. Lu, and S. Wen. “Multistability of state-dependent switching neural networks with discontinuous nonmonotonic piecewise linear activation functions.” In: *Neurocomputing* 437 (May 2021), pp. 300–311. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.01.046. URL: <http://dx.doi.org/10.1016/j.neucom.2021.01.046> (cit. on p. 71).
- [553] A. Goujon, A. Etamadi, and M. Unser. “On the number of regions of piecewise linear neural networks.” In: *Journal of Computational and Applied Mathematics* 441 (May 2024), p. 115667. ISSN: 0377-0427. DOI: 10.1016/j.cam.2023.115667. URL: <http://dx.doi.org/10.1016/j.cam.2023.115667> (cit. on p. 71).
- [554] M. Wang, B. Liu, and H. Foroosh. “Look-Up Table Unit Activation Function for Deep Convolutional Neural Networks.” In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2018. DOI: 10.1109/wacv.2018.00139. URL: <https://doi.org/10.1109/wacv.2018.00139> (cit. on pp. 72–74, 76).
- [555] F. Piazza, A. Uncini, and M. Zenobi. “Neural networks with digital LUT activation functions.” In: *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*. IEEE, 1993. DOI: 10.1109/ijcnn.1993.716806. URL: <https://doi.org/10.1109/ijcnn.1993.716806> (cit. on p. 72).
- [556] S. Fiori. “Hybrid independent component analysis by adaptive LUT activation function neurons.” In: *Neural Networks* 15.1 (Jan. 2002), pp. 85–94. DOI: 10.1016/S0893-6080(01)00105-8. URL: [https://doi.org/10.1016/S0893-6080\(01\)00105-8](https://doi.org/10.1016/S0893-6080(01)00105-8) (cit. on p. 72).
- [557] M. Kang and D. Palmer-Brown. “An Adaptive Function Neural Network (ADFNN) Classifier.” In: *2005 International Conference on Neural Networks and Brain*. IEEE, 2005. DOI: 10.1109/icnnb.2005.1614681. URL: <https://doi.org/10.1109/icnnb.2005.1614681> (cit. on p. 72).
- [558] M. Kang and D. Palmer-Brown. “A Multi-layer ADaptive Function Neural Network (MADFNN) for Letter Image Recognition.” In: *2007 International Joint Conference on Neural Networks*. IEEE, Aug. 2007. DOI: 10.1109/ijcnn.2007.4371406. URL: <https://doi.org/10.1109/ijcnn.2007.4371406> (cit. on p. 72).

- [559] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. "Maxout Networks." In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by S. Dasgupta and D. McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1319–1327. URL: <https://proceedings.mlr.press/v28/goodfellow13.html> (cit. on pp. 72, 73).
- [560] M. S. Hanif and M. Bilal. "Competitive residual neural network for image classification." In: *ICT Express* 6.1 (Mar. 2020), pp. 28–37. DOI: 10.1016/j.ictex.2019.06.001. URL: <https://doi.org/10.1016/j.ictex.2019.06.001> (cit. on p. 73).
- [561] G. Castaneda, P. Morris, and T. M. Khoshgoftaar. "Evaluation of maxout activations in deep learning across several big data domains." In: *Journal of Big Data* 6.1 (Aug. 2019). DOI: 10.1186/s40537-019-0233-0. URL: <https://doi.org/10.1186/s40537-019-0233-0> (cit. on p. 73).
- [562] L. R. Stützel, F. Brieger, H. Finger, S. Füllhase, and G. Pipa. "Adaptive Blending Units: Trainable Activation Functions for Deep Neural Networks." In: *Advances in Intelligent Systems and Computing*. Springer International Publishing, 2020, pp. 37–50. DOI: 10.1007/978-3-030-52243-8_4. URL: https://doi.org/10.1007/978-3-030-52243-8_4 (cit. on pp. 73, 74).
- [563] F. Manessi and A. Rozza. "Learning Combinations of Activation Functions." In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, Aug. 2018. DOI: 10.1109/icpr.2018.8545362. URL: <https://doi.org/10.1109/icpr.2018.8545362> (cit. on p. 73).
- [564] D. Klabjan and M. Harmon. "Activation Ensembles for Deep Neural Networks." In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, Dec. 2019. DOI: 10.1109/bigdata47090.2019.9006069. URL: <https://doi.org/10.1109/bigdata47090.2019.9006069> (cit. on p. 73).
- [565] P. M. Baggenstoss. "Trainable Compound Activation Functions for Machine Learning." In: *2022 30th European Signal Processing Conference (EUSIPCO)*. IEEE, Aug. 2022. DOI: 10.23919/eusipco55093.2022.9909774. URL: <http://dx.doi.org/10.23919/EUSIPCO55093.2022.9909774> (cit. on p. 74).
- [566] P. M. Baggenstoss. "Improved Auto-Encoding Using Deterministic Projected Belief Networks and Compound Activation Functions." In: *2023 31st European Signal Processing Conference (EUSIPCO)*. IEEE, Sept. 2023. DOI: 10.23919/eusipco58844.2023.10290080. URL: <http://dx.doi.org/10.23919/EUSIPCO58844.2023.10290080> (cit. on p. 74).
- [567] Z. Liao. *Trainable Activation Function in Image Classification*. 2020. DOI: 10.48550/ARXIV.2004.13271. URL: <https://arxiv.org/abs/2004.13271> (cit. on pp. 74, 76).
- [568] A. Ismail et al. "Predictions of bridge scour: Application of a feed-forward neural network with an adaptive activation function." In: *Engineering Applications of Artificial Intelligence* 26.5-6 (May 2013), pp. 1540–1549. DOI: 10.1016/j.engappai.2012.12.011. URL: <https://doi.org/10.1016/j.engappai.2012.12.011> (cit. on p. 74).
- [569] A. D. Jagtap, Y. Shin, K. Kawaguchi, and G. E. Karniadakis. "Deep Kronecker neural networks: A general framework for neural networks with adaptive activation functions." In: *Neurocomputing* 468 (Jan. 2022), pp. 165–180. DOI: 10.1016/j.neucom.2021.10.036. URL: <https://doi.org/10.1016/j.neucom.2021.10.036> (cit. on pp. 74, 75).
- [570] M. Goyal, R. Goyal, and B. Lall. *Learning Activation Functions: A new paradigm for understanding Neural Networks*. 2019. DOI: 10.48550/ARXIV.1906.09529. URL: <https://arxiv.org/abs/1906.09529> (cit. on p. 75).
- [571] F. Piazza et al. "Artificial Neural Networks With Adaptive Polynomial Activation Function." In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*. 1992 (cit. on p. 75).
- [572] K.-C. J. Chen and J.-W. Liang. "A Two-stage Training Mechanism for the CNN with Trainable Activation Function." In: *2020 International SoC Design Conference (ISOC)*. IEEE, Oct. 2020. DOI: 10.1109/isoc50952.2020.9333116. URL: <http://dx.doi.org/10.1109/ISOC50952.2020.9333116> (cit. on p. 75).
- [573] T. L. Fonseca and L. Goliatt. "Extreme Learning Machine Based Model Improved with Adaptive Activation Functions." In: *Computational Intelligence in Information Systems*. Springer International Publishing, 2021, pp. 119–128. ISBN: 9783030681333. DOI: 10.1007/978-3-030-68133-3_12. URL: http://dx.doi.org/10.1007/978-3-030-68133-3_12 (cit. on p. 75).
- [574] M. Deepthi, G. N. V. R. Vikram, and P. Venkatappareddy. "Development of a novel activation function based on Chebyshev polynomials: an aid for classification and denoising of images." In: *The Journal of Supercomputing* (June 2023). DOI: 10.1007/s11227-023-05466-y. URL: <https://doi.org/10.1007/s11227-023-05466-y> (cit. on p. 75).
- [575] P. Venkatappareddy, J. Culli, S. Srivastava, and B. Lall. "A Legendre polynomial based activation function: An aid for modeling of max pooling." In: *Digital Signal Processing* 115 (Aug. 2021), p. 103093. DOI: 10.1016/j.dsp.2021.103093. URL: <https://doi.org/10.1016/j.dsp.2021.103093> (cit. on p. 75).
- [576] V. S. Lokhande, S. Tasneeyapant, A. Venkatesh, S. N. Ravi, and V. Singh. "Generating Accurate Pseudo-Labels in Semi-Supervised Learning and Avoiding Overconfident Predictions via Hermite Polynomial Activations." In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. DOI: 10.1109/cvpr42600.2020.01145. URL: <http://dx.doi.org/10.1109/cvpr42600.2020.01145> (cit. on p. 76).
- [577] A. Molina, P. Schramowski, and K. Kersting. "Padé Activation Units: End-to-end Learning of Flexible Activation Functions in Deep Networks." In: *International Conference on Learning Representations*. 2020. URL: <https://openreview.net/forum?id=BJ1BSkHtDS> (cit. on pp. 76, 77).
- [578] C. Brezinski. "Padé Approximations." In: *Computational Aspects of Linear Control*. Springer US, 2002, pp. 87–134. DOI: 10.1007/978-1-4613-0261-2_4. URL: https://doi.org/10.1007/978-1-4613-0261-2_4 (cit. on p. 76).
- [579] C. Brezinski. "Extrapolation algorithms and Padé approximations: a historical survey." In: *Applied Numerical Mathematics* 20.3 (Mar. 1996), pp. 299–318. DOI: 10.1016/0168-9274(95)00110-7. URL: [https://doi.org/10.1016/0168-9274\(95\)00110-7](https://doi.org/10.1016/0168-9274(95)00110-7) (cit. on p. 76).
- [580] Q. Delfosse, P. Schramowski, M. Mundt, A. Molina, and K. Kersting. *Adaptive Rational Activations to Boost Deep Reinforcement Learning*. 2021. DOI: 10.48550/ARXIV.2102.09407. URL: <https://arxiv.org/abs/2102.09407> (cit. on p. 76).
- [581] N. Boulle, Y. Nakatsukasa, and A. Townsend. "Rational neural networks." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 14243–14253. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/a3f390d88e4c41f2747bfa2f1b5f87db-Paper.pdf (cit. on pp. 76, 77).
- [582] Z. Chen, F. Chen, R. Lai, X. Zhang, and C.-T. Lu. "Rational Neural Networks for Approximating Graph Convolution Operator on Jump Discontinuities." In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, Nov. 2018. DOI: 10.1109/icdm.2018.00021. URL: <https://doi.org/10.1109/icdm.2018.00021> (cit. on p. 76).
- [583] M. Trimmel, M. Zanfir, R. Hartley, and C. Sminchisescu. "ERA: Enhanced Rational Activations." In: *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 722–738. DOI: 10.1007/978-3-031-20044-1_41. URL: https://doi.org/10.1007/978-3-031-20044-1_41 (cit. on p. 77).
- [584] K. Biswas, S. Banerjee, and A. K. Pandey. *Orthogonal-Padé Activation Functions: Trainable Activation functions for smooth and faster convergence in deep networks*. 2021. DOI: 10.48550/ARXIV.2106.09693. URL: <https://arxiv.org/abs/2106.09693> (cit. on pp. 77, 78).
- [585] S. Guarnieri et al. "Multilayer feedforward networks with adaptive spline activation function." In: *IEEE Transactions on Neural Networks* 10.3 (May 1999), pp. 672–683. DOI: 10.1109/72.761726. URL: <https://doi.org/10.1109/72.761726> (cit. on p. 77).
- [586] M. Solazzi and A. Uncini. "Artificial neural networks with adaptive multidimensional spline activation functions." In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. IEEE, 2000. DOI: 10.1109/ijcnn.2000.861352. URL: <https://doi.org/10.1109/ijcnn.2000.861352> (cit. on p. 77).
- [587] P. Campolucci, F. Capperelli, S. Guarnieri, F. Piazza, and A. Uncini. "Neural networks with adaptive spline activation function." In: *Proceedings of 8th Mediterranean Electrotechnical Conference on Industrial Applications in Power Systems, Computer Science and Telecommunications (MELECON 96)*. IEEE, 1996. DOI: 10.1109/melcon.1996.551220. URL: <https://doi.org/10.1109/melcon.1996.551220> (cit. on p. 77).

- [588] P. Bohra, J. Campos, H. Gupta, S. Aziznejad, and M. Unser. “Learning Activation Functions in Deep (Spline) Neural Networks.” In: *IEEE Open Journal of Signal Processing* 1 (2020), pp. 295–309. DOI: 10.1109/ojosp.2020.3039379. URL: <https://doi.org/10.1109/ojosp.2020.3039379> (cit. on p. 77).
- [589] S. Lane, M. Flax, D. Handelman, and J. Gelfand. “Multi-Layer Perceptrons with B-Spline Receptive Field Functions.” In: *Advances in Neural Information Processing Systems*. Ed. by R. Lippmann, J. Moody, and D. Touretzky. Vol. 3. Morgan-Kaufmann, 1990. URL: https://proceedings.neurips.cc/paper_files/paper/1990/file/94f6d7e04a4d452035300f18b984988c-Paper.pdf (cit. on p. 77).
- [590] A. Vaicaitis and J. Dooley. “Segmented Spline Curve Neural Network for Low Latency Digital Predistortion of RF Power Amplifiers.” In: *IEEE Transactions on Microwave Theory and Techniques* 70.11 (Nov. 2022), pp. 4910–4915. DOI: 10.1109/tmtt.2022.3210034. URL: <https://doi.org/10.1109/tmtt.2022.3210034> (cit. on p. 77).
- [591] R. G. Kumar and Y. Kumaraswamy. “Spline Activated Neural Network for Classifying Cardiac Arrhythmia.” In: *International Journal of Soft Computing* 9.6 (2014), pp. 377–385. DOI: 10.36478/ijscmp.2014.377.385. URL: <https://medwelljournals.com/abstract/?doi=ijscmp.2014.377.385> (cit. on p. 77).
- [592] H. A. Mayer and R. Schwaiger. “Evolution of Cubic Spline Activation Functions for Artificial Neural Networks.” In: *Progress in Artificial Intelligence*. Springer Berlin Heidelberg, 2001, pp. 63–73. DOI: 10.1007/3-540-45329-6_10. URL: https://doi.org/10.1007/3-540-45329-6_10 (cit. on p. 77).
- [593] M. Solazzi, A. Uncini, and F. Piazza. “Neural equalizer with adaptive multidimensional spline activation functions.” In: *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing (Cat. No.00CH37100)*. IEEE, 2000. DOI: 10.1109/icassp.2000.860155. URL: <https://doi.org/10.1109/icassp.2000.860155> (cit. on p. 77).
- [594] S. Neumayer, A. Goujon, P. Bohra, and M. Unser. “Approximation of Lipschitz Functions Using Deep Spline Neural Networks.” In: *SIAM Journal on Mathematics of Data Science* 5.2 (May 2023), pp. 306–322. ISSN: 2577-0187. DOI: 10.1137/22m1504573. URL: <http://dx.doi.org/10.1137/22m1504573> (cit. on pp. 77, 78).
- [595] S. Ducotterd, A. Goujon, P. Bohra, D. Perdios, S. Neumayer, and M. Unser. *Improving Lipschitz-Constrained Neural Networks by Learning Activation Functions*. 2022. DOI: 10.48550/ARXIV.2210.16222. URL: <https://arxiv.org/abs/2210.16222> (cit. on p. 77).
- [596] S. Aziznejad and M. Unser. “Deep Spline Networks with Control of Lipschitz Regularity.” In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2019. DOI: 10.1109/icassp.2019.8682547. URL: <http://dx.doi.org/10.1109/ICASSP.2019.8682547> (cit. on p. 77).
- [597] D. Fakhoury, E. Fakhoury, and H. Speleers. “ExSpliNet: An interpretable and expressive spline-based neural network.” In: *Neural Networks* 152 (Aug. 2022), pp. 332–346. DOI: 10.1016/j.neunet.2022.04.029. URL: <https://doi.org/10.1016/j.neunet.2022.04.029> (cit. on p. 77).
- [598] A. Uncini. “Sound Synthesis by Flexible Activation Function Recurrent Neural Networks.” In: *Neural Nets*. Springer Berlin Heidelberg, 2002, pp. 168–177. DOI: 10.1007/3-540-45808-5_19. URL: https://doi.org/10.1007/3-540-45808-5_19 (cit. on p. 77).
- [599] N. Kuzuya and T. Nagao. “Designing B-spline-based Highly Efficient Neural Networks for IoT Applications on Edge Platforms.” In: *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Oct. 2022. DOI: 10.1109/smc53654.2022.9945478. URL: <https://doi.org/10.1109/smc53654.2022.9945478> (cit. on p. 77).
- [600] E. López-Rubio, F. Ortega-Zamorano, E. Domínguez, and J. Muñoz-Pérez. “Piecewise Polynomial Activation Functions for Feedforward Neural Networks.” In: *Neural Processing Letters* 50.1 (Jan. 2019), pp. 121–147. DOI: 10.1007/s11063-018-09974-4. URL: <https://doi.org/10.1007/s11063-018-09974-4> (cit. on pp. 77, 78).
- [601] Q. Su, x. Liao, and L. Carin. “A Probabilistic Framework for Nonlinearities in Stochastic Neural Networks.” In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/35936504a37d53e03abdfbc7318d9ec7-Paper.pdf (cit. on p. 78).
- [602] J. T. Barron. *Squareplus: A Softplus-Like Algebraic Rectifier*. 2021. DOI: 10.48550/ARXIV.2112.11687. URL: <https://arxiv.org/abs/2112.11687> (cit. on pp. 78, 79).
- [603] J.-R. Chang and Y.-S. Chen. *Batch-normalized Maxout Network in Network*. 2015. DOI: 10.48550/ARXIV.1511.02583. URL: <https://arxiv.org/abs/1511.02583> (cit. on p. 81).
- [604] M. Wang, B. Liu, and H. Foroosh. “Wide Hidden Expansion Layer for Deep Convolutional Neural Networks.” In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2020. DOI: 10.1109/wacv45572.2020.9093436. URL: <https://doi.org/10.1109/wacv45572.2020.9093436> (cit. on pp. 81, 82).
- [605] C. Eisenach, Z. Wang, and H. Liu. “Nonparametrically Learning Activation Functions in Deep Neural Nets.” In: *International Conference on Learning Representations Workshops*. 2017. URL: <https://openreview.net/forum?id=H1wgawqxl> (cit. on p. 81).
- [606] C. J. Vercellino and W. Y. Wang. “Hyperactivations for Activation Function Exploration.” In: *Proceedings of the 3rd International Conference on Neural Information Processing Systems. NIPS'17*. Long Beach, California, USA, 2017. URL: http://metalearning.ml/2017/papers/metalearn17_vercellino.pdf (cit. on pp. 81, 82).
- [607] S. Zhang, Q. Liu, X. Wu, and W. Chen. “A Self-Adaptive and Multiple Activation Function Neural Network for Facial Expression Recognition.” In: *Proceedings of the 2021 5th International Conference on Electronic Information Technology and Computer Engineering. EITCE 2021*. ACM, Oct. 2021. DOI: 10.1145/3501409.3501605. URL: <http://dx.doi.org/10.1145/3501409.3501605> (cit. on p. 81).
- [608] I. Castelli and E. Trentin. “Semi-supervised Weighted Maximum-Likelihood Estimation of Joint Densities for the Co-training of Adaptive Activation Functions.” In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 62–71. DOI: 10.1007/978-3-642-28258-4_7. URL: https://doi.org/10.1007/978-3-642-28258-4_7 (cit. on p. 81).
- [609] I. Castelli and E. Trentin. “Supervised and Unsupervised Co-training of Adaptive Activation Functions in Neural Nets.” In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 52–61. DOI: 10.1007/978-3-642-28258-4_6. URL: https://doi.org/10.1007/978-3-642-28258-4_6 (cit. on p. 81).
- [610] I. Castelli and E. Trentin. “Combination of supervised and unsupervised learning for training the activation functions of neural networks.” In: *Pattern Recognition Letters* 37 (Feb. 2014), pp. 178–191. DOI: 10.1016/j.patrec.2013.06.013. URL: <https://doi.org/10.1016/j.patrec.2013.06.013> (cit. on p. 81).
- [611] A. Apicella, F. Isgrò, and R. Prevete. “A simple and efficient architecture for trainable activation functions.” In: *Neurocomputing* 370 (Dec. 2019), pp. 1–15. DOI: 10.1016/j.neucom.2019.08.065. URL: <https://doi.org/10.1016/j.neucom.2019.08.065> (cit. on pp. 81, 82).
- [612] F. u. A. A. Minhas and A. Asif. *Learning Neural Activations*. 2019. DOI: 10.48550/ARXIV.1912.12187. URL: <https://arxiv.org/abs/1912.12187> (cit. on p. 82).
- [613] H. Klopries and A. Schwung. “Flexible Activation Bag: Learning Activation Functions in Autoencoder Networks.” In: *2023 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, Apr. 2023. DOI: 10.1109/icit58465.2023.10143113. URL: <http://dx.doi.org/10.1109/ICIT58465.2023.10143113> (cit. on p. 82).
- [614] E. Jang, S. Gu, and B. Poole. “Categorical Reparameterization with Gumbel-Softmax.” In: *International Conference on Learning Representations*. 2017. URL: <https://openreview.net/forum?id=rkE3y85ee> (cit. on p. 82).
- [615] Ö. F. Ertuğrul. “A novel type of activation function in artificial neural networks: Trained activation function.” In: *Neural Networks* 99 (Mar. 2018), pp. 148–157. DOI: 10.1016/j.neunet.2018.01.007. URL: <https://doi.org/10.1016/j.neunet.2018.01.007> (cit. on p. 82).
- [616] S. Scardapane, S. V. Vaerenbergh, S. Totaro, and A. Uncini. “Kafnets: Kernel-based non-parametric activation functions for neural networks.” In: *Neural Networks* 110 (Feb. 2019), pp. 19–32. DOI: 10.1016/j.neunet.2018.11.002. URL: <https://doi.org/10.1016/j.neunet.2018.11.002> (cit. on p. 83).

- [617] S. Kiliçarslan and M. Celik. “KAF+RSigELU: a nonlinear and kernel-based activation function for deep neural networks.” In: *Neural Computing and Applications* 34.16 (Apr. 2022), pp. 13909–13923. DOI: 10.1007/s00521-022-07211-7. URL: <https://doi.org/10.1007/s00521-022-07211-7> (cit. on p. 83).
- [618] W. Zhang, Z. Han, X. Chen, B. Liu, H. Jia, and Y. Tang. “Fully Kerneted Neural Networks.” In: *Journal of Mathematics* 2023 (June 2023). Ed. by Q. Wu, pp. 1–9. DOI: 10.1155/2023/1539436. URL: <https://doi.org/10.1155/2023/1539436> (cit. on p. 83).
- [619] S. Brad. “Enhancing Creativity in Deep Learning Models with SAVE-Inspired Activation Functions.” In: *Towards AI-Aided Invention and Innovation*. Springer Nature Switzerland, 2023, pp. 147–171. ISBN: 9783031425325. DOI: 10.1007/978-3-031-42532-5_12. URL: http://dx.doi.org/10.1007/978-3-031-42532-5_12 (cit. on pp. 83, 84).
- [620] S. Brad and E. Şetco. “An Interactive Artificial Intelligence System for Inventive Problem-Solving.” In: *Systematic Innovation Partnerships with Artificial Intelligence and Information Technology*. Springer International Publishing, 2022, pp. 165–177. ISBN: 9783031172885. DOI: 10.1007/978-3-031-17288-5_15. URL: http://dx.doi.org/10.1007/978-3-031-17288-5_15 (cit. on p. 83).