ISSN1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20230614000735
Original scientific paper

The Adaptive Quadratic Linear Unit (AQuLU): Adaptive Non Monotonic Piecewise Activation Function

Zhandong WU, Haiye YU, Lei ZHANG*, Yuanyuan SUI

Abstract: The activation function plays a key role in influencing the performance and training dynamics of neural networks. There are hundreds of activation functions widely used as rectified linear units (ReLUs), but most of them are applied to complex and large neural networks, which often have gradient explosion and vanishing gradient problems. By studying a variety of non-monotonic activation functions, we propose a method to construct a non-monotonic activation function, $x \cdot \Phi(x)$, with $\Phi(x) \in [0, 1]$. With the hardening treatment of $\Phi(x)$, we propose an adaptive non-monotonic segmented activation function, called the adaptive quadratic linear unit, abbreviated as AQuLU, which ensures the sparsity of the input data and improves training efficiency. In image classification based on different state-of-the-art neural network architectures, the performance of AQuLUs has significant advantages for more complex and deeper architectures with various activation functions. The ablation experimental study further validates the compatibility and stability of AQuLUs with different depths, complexities, optimizers, learning rates, and batch sizes. We thus demonstrate the high efficiency, robustness, and simplicity of AQuLUs.

Keywords: activation function; AQuLU; CaLU; deep learning; ExpExpish; LogLogish; LaLU

1 INTRODUCTION

In recent years, the pace at which AI technologies are changing has been dramatic. It has been the rapid emergence of deep learning technology that has not only led to the generation of endless neural network models but also greatly improved computing power to collect and share large quantities of training data. The contribution of activation functions is essential in the development of neural networks and plays an important role in nonlinear mapping representations between input data and output data from deep neural networks. Carefully selected activation functions (AFs) not only accelerate gradient convergence but also enhance the generalization of the network [1], and AFs themselves affect the expressiveness of the neural network, that is, the ability of the network to approximate the objective function [2].

Early saturation AFs such as sigmoid [3] and tanh functions can express how biological neural networks work well, but in deep neural networks, saturation AFs are prone to gradients that disappear and are limited by different obstacles when training deep networks. Rectified linear units (ReLUs) [4] are currently the most commonly used AFs and initially solve the gradient disappearance problem because they extract sparse features and run efficiently [5, 6], but they have no upper bound, no differentiation at 0, positive average activation [7] and "dying" issues [8]. To solve the negative use problem of ReLUs, an AF based on linear [9] and nonlinear [7] improvement is proposed. A ReLU forces the negative value of the input to and replaces it with a linear or nonlinear function, allowing the negative input to participate in learning. The unbounded output of a ReLU may lead to instability, and in addition to embedded systems with dedicated hardware, bounded AFs [24] are needed. To enhance the robustness of ReLUs, their positive region [26] is suitably modified.

A ReLU, dropout [32], and zoneout [33] increase the sparseness of output by multiplying by 1 or 0 according to their own needs, making most problems linearly divisible and interpretable and require less storage. GELUs [34] are obtained by combining dropout, zoneout and ReLU properties, and then, a variety of AFs [35] satisfying $x \cdot \Phi(x)$ are born, all of which are non-monotonic AFs and AFs with small negative output and sparse output, where $\Phi(x)$ is a gate control. $\Phi(x)$ can also be a random regularizer, and $x \cdot \Phi(x)$ represents the expectation of the random regularizer on the input x.

AFs can be saturated or unsaturated and can have parameters that include hyper parameters for human intervention and parameters that are variable as training proceeds in the learning process. A parameter-learnable AF is called an adaptive AF. The adaptive activation function improves the fitting performance of the network and enhances the learning performance by adding adjustable parameters so that the gradient can be effectively transferred between the network layers. Moreover, the consistency of the network model can be improved, and the feature selection ability of the model can be improved at the same time. Maxout [15] is considered an earlier adaptive unsaturated AF. Adaptive AFs [44] have obvious advantages. Neural networks are complex systems with different sizes of data activated per layer and even per channel, while adaptive AFs cater to the uncertain nature of neural networks. Thus far, there are nearly a hundred AFs, but the best AFs have not been obtained. While the best AF is recognized, AFs generally need to meet the following criteria:

- AFs are unsaturated, continuous, and smooth, and are nonlinear functions that are almost differentiable everywhere.
- AFs are preferably close to identity mapping, making network training more stable and gradients easier to reverse propagation and simpler to derive.
- AFs are simple to calculate and have few parameters, which can improve the network computing efficiency. Both positive and negative regions can be activated to control output with an average activation value of 0 for faster learning.

In this work, while the above activation features have had great success in some deep learning networks, the classification results are not ideal in more complex networks in our research. Network efficiency is very important. Reducing training time while getting better performance, we study a sparse activation function from here, and can learn to obtain better network performance.

To improve the accuracy of the classification results, we combine the properties of $x\cdot\Phi(x)$, with $\Phi(x)\in[0,\,1]$, with those of adaptive functions and propose an adaptive non-monotonic AF. Non-monotonic functions output small negative values that push the average activation value to 0, accelerating convergence in model training [7]. Sparse output can be increased to help information disentanglement, efficient variable-size representation, linear separability and distribution [5]. Adaptive AFs have been proven to be very effective for training networks, which not only improves the convergence of the network but also reduces the loss function faster and can accelerate the minimization of the training loss value [56]. The sparsity of the current adaptive activation function is not good, but there are too many parameters with good sparsity, so few parameters are not used to solve the sparsity and better fitting of the network. More generally, we propose AFs that can fit arbitrary non-monotonic AFs with gating mechanisms. They are represented as follows:

$$f(x)=x\cdot\Phi(x)$$
$$=x\cdot\begin{cases}1 & \text{if } x\geq\dfrac{1-\beta}{\alpha}\\ \alpha x+\beta & \text{if } -\dfrac{\beta}{\alpha}\leq x<\dfrac{1-\beta}{\alpha}\\ 0 & \text{if } x<-\dfrac{\beta}{\alpha}\end{cases}\tag{1}$$

α and β are Hyper-parameters. This is called the quadratic linear unit (QuLU). If the parameters are trainable, we propose the adaptive quadratic linear unit (AQuLU). The QuLU is the same as the ReLU when $\alpha\to+\infty,\beta\to1$. When $\alpha=\dfrac{1}{6},\beta=0.5$, the QuLU equals HardSwish [59]. When $\alpha=0.5,\beta=1$, the QuLU is the same as SqREU [61]. The AQuLU is a piecewise function, and the specific sub-regions need to be obtained in self-learning.

The AQuLU combines the advantages of a ReLU. Moreover, AQuLU can automatically learn suitable parameters for deeper networks, which not only prevents gradient disappearance and explosion but also improves training efficiency through the sparsity of input data. We demonstrate efficiency, robustness and simplicity on various datasets and networks where AQuLU outperforms other non-monotonic AFs.

The organizational structure of this article is as follows. Related work is presented in Section 2. In Sections 3 and 4, we explain the origin of the AQuLU by describing its definitions and properties and how parameters are defined. In Section 5, we demonstrate the efficiency and robustness of AQuLUs on various datasets and networks and demonstrate the results. Section 6 presents conclusions for the whole work.

2 RELATED WORK

2.1 Rectified Activation Functions

A ReLU [4] is a very simple function. When the input is positive, it is an identity function, and using for negative input yields the following:

$$ReLU(x)=x\cdot\begin{cases}1, & x\geq0\\ 0, & x<0\end{cases}=\begin{cases}x, & x\geq0\\ 0, & x<0\end{cases}\tag{2}$$

Therefore, the range of the ReLU is $[0,\infty)$. The gradients for positive and negative inputs are 1 and 0, respectively. The ReLU function solves the vanishing gradient problem of saturation functions when the input is positive. Although there is no gradient upgrade due to the activation value being 0 when the input is negative, ReLUs are already widely used for deep learning models.

To solve the "Dying-ReLU" problem, the function of the negative region is transformed. Leaky ReLU [9], PReLU [10], and RReLU [11] replace 0 with a linear function with a smaller slope. The slope of Leaky ReLU is a super-parameter, the slope of PReLU is a trainable parameter, and the slope of RReLU comes from a uniform distribution. SReLU [62] is a novel S-shaped rectified linear unit that learns convex and nonconvex functions with four learnable parameters. When selecting different parameters, SReLU can degenerate to ReLU, LReLU, and PReLU. To prevent the learned CNN from being likely to be sensitive to the tiny jitter of nonlinear activation inputs, it is prone to overfitting. The randomly translational nonlinear AFs RTReLU and RTPReLU [13] are proposed for deep CNNs. RTReLU is proposed for ReLU translation in the *X*-axis direction, and *L**ReLU [63] is proposed for *Y*-axis translation. To prevent gradient disappearance, the LiSA [64] of the sigmoid linear fit and the PLU [65] of the tanh fit are provided. However, neither are saturated functions, which are linear functions with small slopes in both positive and negative regions.

ISRLUs [15], TaLUs [66], SineReLUs [67], and SignReLUs [16] introduce nonlinear functions to replace output with values of 0 in ReLUs. An exponential linear unit (ELU) [7] uses the exponential function instead of the negative region function in a ReLU, which is expressed as follows:

$$ELU(x)=\begin{cases}x, & x>0\\ \alpha\times\left(e^{x}-1\right), & x\leq0\end{cases}\tag{3}$$

where α is a learnable parameter. The ELU is differentiable, is saturated with large negative inputs, and reduces bias drift. The negative saturation mechanism of an ELU increases the robustness of noise relative to the Leaky ReLU and the PReLU. The ELU is extended to the scaled ELU (SELU) [17] using a scale hyper-parameter to make a positive input with a slope greater than 1. To increase commonality and adaptability, many scholars extend ELUs, including PELU [18], CELU [19], MPELU [20], PREU [68], FELU [21], EELU [22], and PDELU [23]. These methods tend to design a smooth piecewise function and force it to be close to zero. However, the output of the negative parts of these AFs loses ReLU sparseness while adding more parameters. As a result, none of them are as widely used as ReLUs.

2.2 Activation Functions Based on the Squashing Function

AFs are approximately equal to the identity function near 0, making the network more stable while the gradient is easier to return and the derivation is simpler. Therefore,

the scale of the input is controlled using a function called the squashing function ($\Phi(x)$). The squashing function (SF) is also equivalent to a gate function, which does not directly control the opening and closing of a gate (the SF of a ReLU is a step function, and the gate is either open or closed) but controls the opening degree of the gate. The first proposal based on the SF is the Gaussian error linear unit (GELU) [34], whose SF is the cumulative distribution function of a Gaussian function. The GELU is also extended to the symmetrical Gaussian error linear unit (SGELU) [69]. Although symmetric functions can achieve faster convergence, learning can be very slow if the weights are too small [70]. The SiLU [34] is born when a sigmoid function is used as the squeeze function. The SiLU is parameterized to form Swish [35]. To increase the slope of the positive region of the SiLU curve, E-Swish [42] is proposed. Richard's curve is a generalized logical curve that allows for more flexibility. In general, sigmoid rotates around the inflection point symmetrically. Richard's curve exhibits a rotational asymmetric S-shape.

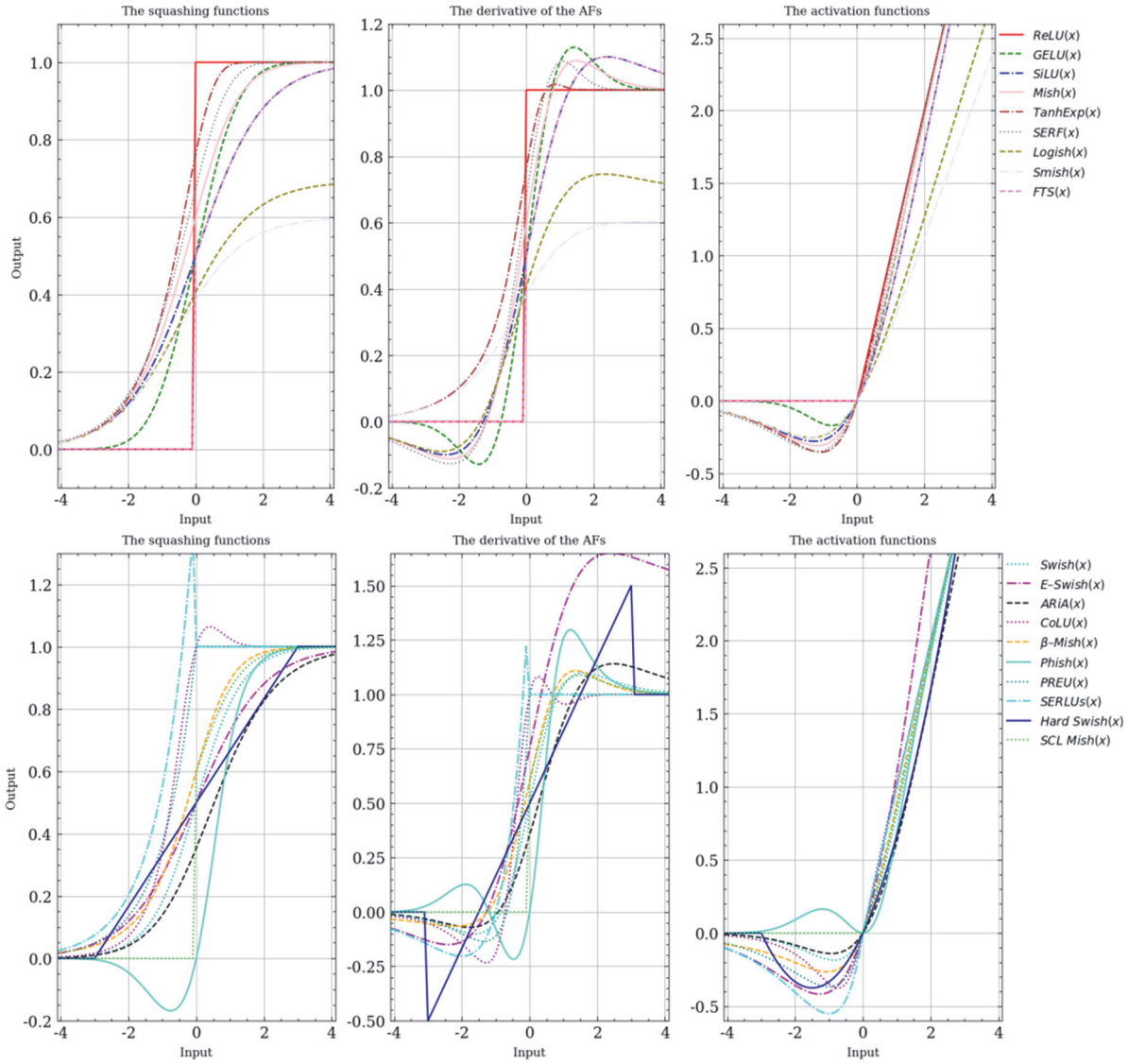


Figure 1 Curves for AFs and their derivatives and squeezing functions

By limiting the number of hyper-parameters, $A = 0$, and the other parameters are 1. Only parameters B and v are changed. ARiA [41] is proposed. Based on sigmoid, the revision proposes a collapsing linear unit (CoLU) [72]. If tanh is combined with Softplus [5] to form a new SF, a novel self-regular non-monotonic Mish [36] is proposed. Mish is parameterized to form β-Mish [74].

The SF is a combination of basic elementary functions, so various AFs based on the SF are born, as shown in Tab. 1. A curve of the SF and its formed AF is shown in Fig. 1. As shown on the left, the saturation value of Smish and Logish in the positive region is not 1, and CoLU, SERLUs, and Phish are not strict SFs. Other SFs with $\Phi(0)$ are different when $x = 0$ and the saturation rates are different. As shown on the right, the shapes of the different AFs are basically the same, but the details vary.

First, when $x \geq 0$, the distance between the AFs and the identity map varies. Next, except for Phish, when $x < 0$, the other AFs generate a smaller negative value, but the minimum value and the speed of saturation are different. However, it is difficult to determine which of so many AFs is the best, but this type of function has many advantages, such as nonlinearity, differentiation, $f(x) \approx x$, an appropriate gradient, and smaller negative output.

Table 1 Mathematical formulas and derivatives of various activation functions

AFs	Squashing State Function	State Function	Derivative State Function
ReLU [4]	$\Phi(x)=\begin{cases}1, & x\geq 0\\ 0, & x<0\end{cases}$	$f(x)=\begin{cases}x, & x\geq 0\\ 0, & x<0\end{cases}$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\begin{cases}1, & x\geq 0\\ 0, & x<0\end{cases}$
GELU [34]	$\Phi(x)=\frac{1}{2}\left[1+erf\left(x/\sqrt{2}\right)\right]$	$f(x)=x\cdot\frac{1}{2}\left[1+\mathrm{erf}\left(x/\sqrt{2}\right)\right]$	$\frac{df(x)}{\mathrm{d}x}=x\cdot\frac{1}{\sqrt{2\pi}}\mathrm{e}^{\left(-\frac{x^2}{2}\right)}+\frac{1}{2}\left[1+\mathrm{erf}\left(x/\sqrt{2}\right)\right]$
SiLU [34]	$\Phi(x)=\sigma(x)=(1+\mathrm{e}^{-x})^{-1}$	$f(x)=x\cdot\sigma(x)$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\sigma(x)+f(x)\left(1-\sigma(x)\right)$
Swish [34]	$\Phi(x)=\sigma(x)=(1+\mathrm{e}^{-x})^{-1}$	$f(x)=x\cdot\sigma(\beta x)$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\sigma(\beta x)+\beta f(x)\left(1-\sigma(\beta x)\right)$
E-Swish	$\Phi(x)=\sigma(x)=(1+\mathrm{e}^{-x})^{-1}$	$f(x)=\beta x\cdot\sigma(x)$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\beta\sigma(x)+f(x)\left(1-\sigma(x)\right)$
ARiA [34]	$\Phi(x)=(\sigma(\beta x))^{\alpha}=(1+\mathrm{e}^{-\beta x})^{-\alpha}$	$f(x)=x\cdot(\sigma(\beta x))^{\alpha}$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=(\sigma(\beta x))^{\alpha}+\alpha\beta f(x)\left(1-\sigma(\beta x)\right)$
CoLU [34]	$\Phi(x)=(1-x\mathrm{e}^{-\left(x+\mathrm{e}^{x}\right)})^{-1}$	$f(x)=x\cdot(1-x\mathrm{e}^{-\left(x+\mathrm{e}^{x}\right)})^{-1}$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\frac{\lambda\left(\lambda-x^2\left(\mathrm{e}^{x}+1\right)\right)}{(\lambda-x)^2},\lambda=\mathrm{e}^{\mathrm{e}^{x}+x}$
Mish [34]	$\Phi(x)=\tanh\left(softplus(x)\right),$ $softplus(x)=\ln\left(1+\mathrm{e}^{x}\right)$	$f(x)=x\cdot\tanh\left(softplus(x)\right)$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=x\sigma(x)\cdot\mathrm{sech}^2\left(softplus(x)\right)+\Phi(x)$
β-Mish[34]	$\Phi(x)=\tanh\left(softplus(r)\right),$ $r=\frac{\alpha x}{\sqrt{\beta+x^2}}$	$f(x)=x\cdot\tanh\left(softplus(r)\right)$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\frac{\frac{\alpha\beta x\sqrt{\beta+x^2 r}}{\cosh^2\left(softplus(x)\right)}+(x^2+\beta)^2(1+r)\tanh\left(softplus(x)\right)}{(x^2+\beta)^2(1+r)}$
TanhExp [34]	$\Phi(x)=\tanh\left(\mathrm{e}^{x}\right)$	$f(x)=x\cdot\tanh\left(\mathrm{e}^{x}\right)$	$\frac{df(x)}{dx}=\Phi(x)+\mathrm{e}^{x}\left(x-f(x)\Phi(x)\right)$
SERF [34]	$\Phi(x)=\mathrm{erf}\left(softplus(x)\right)$	$f(x)=x\cdot\mathrm{erf}\left(softplus(x)\right)$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\frac{2}{\sqrt{\pi}}\mathrm{e}^{-\ln\left(\left(1+\mathrm{e}^{x}\right)\right)^2}x\sigma(x)+\Phi(x)$
Logish [34]	$\Phi(x)=\ln\left(1+\sigma(x)\right)$	$f(x)=x\cdot\ln\left(1+\sigma(x)\right)$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\Phi(x)+\frac{x\cdot\mathrm{e}^{-x}(\sigma(x))^2}{1+\sigma(x)}$
Smish [34]	$\Phi(x)=\tanh\left(\ln\left(1+\sigma(x)\right)\right)$	$f(x)=x\cdot\tanh\left(\ln\left(1+\sigma(x)\right)\right)$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\Phi(x)+\left(x-f(x)\Phi(x)\right)\frac{\mathrm{e}^{-x}(\sigma(x))^2}{1+\sigma(x)}$
Phish [34]	$\Phi(x)=\tanh\left(GELU(x)\right)$	$f(x)=x\cdot\tanh\left(GELU(x)\right)$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\Phi(x)+x\cdot\mathrm{sech}^2\left(GELU(x)\right)\frac{\mathrm{d}GELU(x)}{\mathrm{d}x}$
FTS [34]	$\Phi(x)=\begin{cases}\sigma(x), & x\geq 0\\ 0, & x<0\end{cases}$	$f(x)=\begin{cases}SiLU(x), & x\geq 0\\ 0, & x<0\end{cases}$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\begin{cases}\frac{\mathrm{d}SiLU(x)}{\mathrm{d}x}, & x\geq 0\\ 0, & x<0\end{cases}$
PREU [34]	$\Phi(x)=\begin{cases}1, & x\geq 0\\ \mathrm{e}^{\beta x}, & x<0\end{cases}$	$f(x)=\alpha x\cdot\begin{cases}1, & x\geq 0\\ \mathrm{e}^{\beta x}, & x<0\end{cases}$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\begin{cases}\alpha, & x\geq 0\\ \alpha(1+\beta x)\cdot\mathrm{e}^{\beta x}, & x<0\end{cases}$
SERLUs [34]	$\Phi(x)=\begin{cases}1, & x\geq 0\\ \alpha\mathrm{e}^{x}, & x<0\end{cases}$	$f(x)=\lambda x\cdot\begin{cases}1, & x\geq 0\\ \alpha\mathrm{e}^{x}, & x<0\end{cases}$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\begin{cases}\lambda, & x\geq 0\\ \lambda\alpha\mathrm{e}^{x}(1+x), & x<0\end{cases}$
Hard Swish [34]	$\Phi(x)=\begin{cases}0, & x\leq -3\\ 1, & x\geq 3\\ \frac{x}{6}+\frac{1}{2}, & \text{otherwise}\end{cases}$	$f(x)=x\cdot\begin{cases}0, & x\leq -3\\ 1, & x\geq 3\\ \frac{x}{6}+\frac{1}{2}, & \text{otherwise}\end{cases}$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\begin{cases}0, & x\leq -3\\ 1, & x\geq 3\\ \frac{x}{3}+\frac{1}{2}, & \text{otherwise}\end{cases}$
SCL Mish [34]	$\Phi(x)=\begin{cases}\tanh\left(softplus(x)\right), & x\geq 0\\ 0, & x<0\end{cases}$	$f(x)=\begin{cases}Mish(x), & x\geq 0\\ 0, & x<0\end{cases}$	$\frac{\mathrm{d}f(x)}{\mathrm{d}x}=\begin{cases}\frac{dMish(x)}{dx}, & x\geq 0\\ 0, & x<0\end{cases}$

2.3 Learning/Adaptive Activation Functions

Adaptive AFs search for a good shape using knowledge given by the training data [78]; some are parameterized versions of proposed AFs, and some are defined by mixing several distinct functions, with the common mixing method being linear combinations. Most of the aforementioned AFs are proposed and might not be adjusted for the complexity of the dataset. Some of the previously mentioned AFs are also adaptive, such as PReLU [10], SReLU [62], PLU [65], PELU [18], MPELU [20], PREU [68], EELU [22], PDELU [23], Swish [35], and PREU [68] AFs.

The parameterization of sigmoid gave birth to the first adaptive AF AGSig [79] and the sigmoid selector [80] proposed later. In Maxout [43] k nodes are added between

layers to find maximum activation as an output activation value, which is difficult to apply because of the large number of parameters. Adaptive piecewise linear (APL) [44] is defined as a sum of hinge-shape functions, which is actually a combination of a ReLU and other linear functions, and the number of hinges determines the complexity of APL. SPLASH [47] symmetrically handles hinge nodes, with more parameters to learn than APL. The adaptive AF (AAF) [81] was designed by combining the PReLU and PELU using learnable weights. The AAF is costly, as multiple AFs are involved. In [82], different combinations of a ReLU, an identity function, and tanh are learned automatically. A parameterized piecewise function composed of a ReLU, sigmoid and ELU forms RSigELU [83]. Activation sets are used at each layer in [84], and the contribution of each AF is controlled by the trainable weights. Similarly, the self-learning AF (SLAF) [51] calculates the sum of the different functions in a set with learning coefficients. The SLAF can be expressed as follows:

$$SLAF\left(x\right)=\sum_{i=0}^{N-1}\alpha_i\times x^i \tag{4}$$

where α_i is the trainable parameter. Adaptive and trainable AFs are recent trends in adjusting nonlinearity based on data and network complexity. However, the minimal burden is increased in terms of the increased number of parameters.

3 METHODS

Generally, the AF used in fully connected or convolutional neural networks is best for identity transformation. In this way, the output amplitude does not increase significantly as the depth increases so that the network is more stable and the gradient is easier to backpropagate. Regardless of whether it is an attention mechanism [85, 86] or gate control [87], the input is randomized and regularized, so we use this idea to create a general expression for the AF, which is specifically described by the following:

$$g\left(x\right)=x\cdot\Phi\left(x\right) \tag{5}$$

where $\Phi\left(x\right)$ is called the squashing function, $x\in\mathbb{R}^n$, and $\Phi_i\left(x\right)\in\left(0,1\right)$. This approach allows for the design of various AFs.

3.1 Design of the Activation Function Based on the Squashing Function

The design of the SF in GELU has no definite theoretical support, only because the cumulative distribution function (CDF) of the normal distribution compresses the input to (0, 1). In fact, CDFs such as Cauchy and Laplace distributions can also compress the input to (0, 1). Therefore, the AF can also be designed according to the CDFs of the Cauchy and Laplace distributions.

3.1.1 CaLU

When the SF is the standard Cauchy cumulative distribution function, the constructed AF is named CaLU, which is described by

$$\begin{aligned}CaLU\left(x\right)&=x\cdot P\left(C\le x\right)\\&=x\cdot\Phi_{CaLU}\left(x\right)\\&=x\cdot\left(\frac{\arctan\left(x\right)}{\pi}+\frac{1}{2}\right)\end{aligned} \tag{6}$$

where $C\sim Cauchy\left(0,1\right)$ and the derivative of CaLU can be calculated by

$$\frac{\mathrm{d}CaLU\left(x\right)}{\mathrm{d}x}=\frac{1}{\pi}\cdot\frac{x}{x^2+1}+\frac{\arctan\left(x\right)}{\pi}+\frac{1}{2} \tag{7}$$

As seen in Fig. 2a, the range of Φ_{CaLU} is (0, 1). $CaLU\left(x\right)$ is a monotonically increasing function, which is close to identity mapping (below $y = x$) in the region of $x\ge0$ and constant function $\left(y=-\frac{1}{\pi}\right)$ in $x<0$. Overall, the $CaLU\left(x\right)$ image is similar to the ELU [7], SELU [17] and CELU [19] images, but $CaLU\left(x\right)$ is continuous rather than piecewise. There is a derivative in the region $\left[-2,+\infty\right)$ that is close to 0 in $\left(-\infty,-2\right)$. In addition, $CaLU\left(x\right)$ is a multiorder derivative smooth function.

3.1.2 LaLU

When the SF is the standard Laplace cumulative distribution function, the constructed AF is named LaLU, which is defined by

$$\begin{aligned}LaLU\left(x\right)&=x\cdot P\left(L\le x\right)\\&=x\cdot\Phi_{LaLU}\left(x\right)\\&=x\cdot\begin{cases}1-\frac{1}{2}\mathrm{e}^{-x} & \text{if } x\ge0\\\frac{1}{2}\mathrm{e}^{x} & \text{if } x<0\end{cases}\end{aligned} \tag{8}$$

where $L\sim Laplace\left(0,1\right)$ and the derivative of CaLU can be represented by

$$\frac{\mathrm{d}LaLU\left(x\right)}{\mathrm{d}x}=x\cdot\begin{cases}1-\frac{1}{2}\mathrm{e}^{-x}\left(x-1\right) & \text{if } x\ge0\\\frac{1}{2}\mathrm{e}^{x}\left(x+1\right) & \text{if } x<0\end{cases} \tag{9}$$

As seen in Fig. 2b, the range of Φ_{LaLu} is (0, 1), and $LaLU\left(x\right)$ is a nonmonotonic function, which decreases first and then increases from left to right. The curve on the positive semi-axis is basically coincident with $y = x$, and

the minimum value is taken at $x = -1$. Thus, it is an unsaturated function with a range of $(-0.1839, +\infty)$. The curves of LaLU are similar to those of Swish and GELU. There is a derivative in the region $[-6, +\infty)$ that is close to 0 in $(-\infty, -6)$. $LaLU(x)$ is a multiorder derivative smooth function.

3.1.3 LogLogish

LogLog [88] has a value range of (0, 1); it has the potential to replace the classic sigmoid function, the saturated speed is higher, and the null point is higher than 0.5. It can be regarded as a kind of SF whose formula is

$$\Phi_{LogLogish}(x) = 1 - e^{-e^{x}} \tag{10}$$

and the AF based on LogLog can be named Loglogish, which can be represented by

$$LogLogish(x) = x \cdot \Phi_{LogLogish}(x) = x \cdot \left(1 - e^{-e^{x}}\right) \tag{11}$$

The derivative formula of LogLogish can be calculated by

$$\frac{dLogLogish(x)}{dx} = e^{-e^{x}} \cdot \left(x \cdot e^{x} - 1\right) + 1 \tag{12}$$

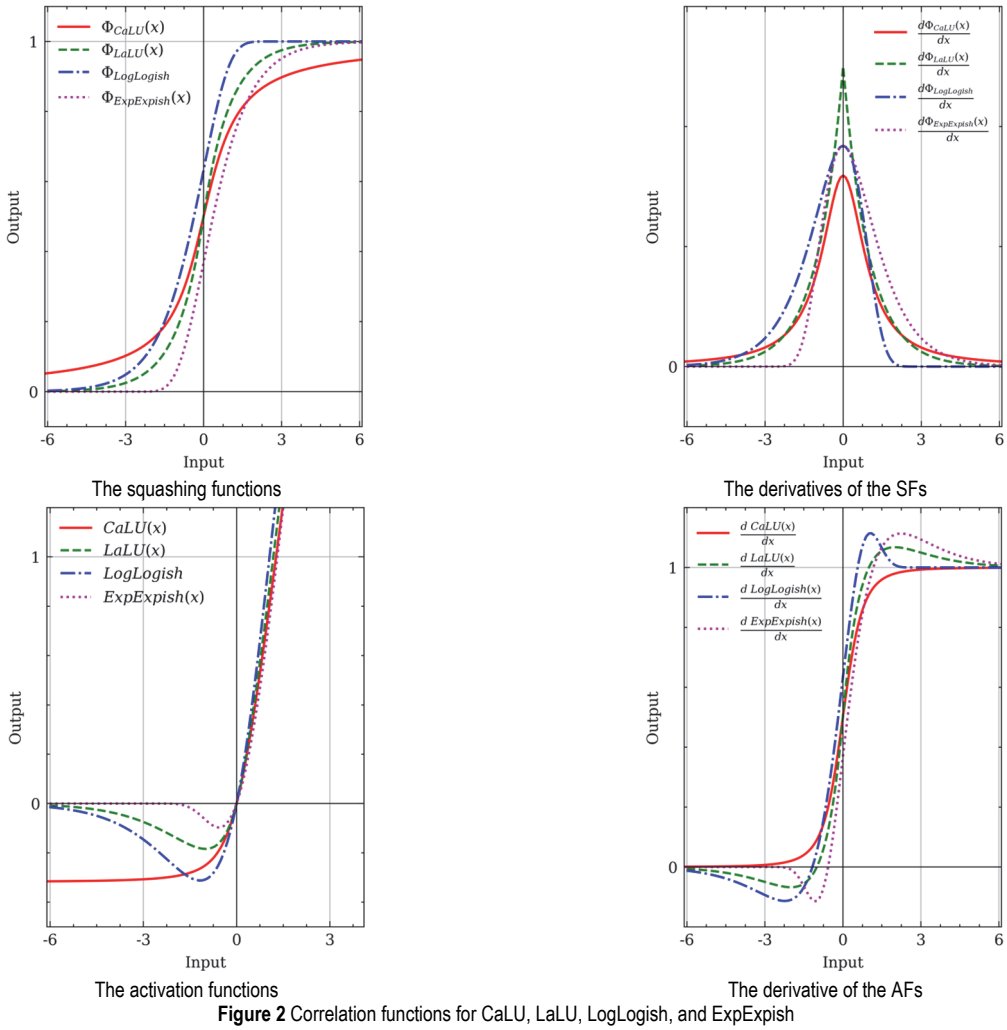


Figure 2 Correlation functions for CaLU, LaLU, LogLogish, and ExpExpish

where $\Phi_{LogLogish}(x) \in (0,1)$ and $LogLogish(x)$ is a non-monotonic function, which decreases first and then increases from left to right. The curve with the positive semiaxis is basically coincident with $y = x$, and the minimum value is taken at $x \approx -1.1722$. Thus, it is an unsaturated function with a range of $(-0.3122, +\infty)$. As seen in Fig. 2c, the curve of $LogLogish(x)$ is similar to that of Mish and closer to the identity map. The curve of $\frac{dLogLogish(x)}{dx}$ shows that the derivative is close to 0 in the $(-\infty, -7]$ region, which increases the sparsity of the output, and there is a derivative in the $(-7, 0)$ region to ensure that the neuron will not "die". In addition, $LogLogish(x)$ is a multiorder derivative smooth function.

3.1.4 ExpExpish

We propose a new function, named ExpExp, which is described by

$$ExpExp(x) = e^{-e^{-x}} \tag{13}$$

ExpExp is a saturated function. When $x \to +\infty$, the function value tends toward 1, and when $x \to -\infty$, the function value tends toward 0. We use ExpExp as the squeezing function, and the AF is named ExpExpish, which is shown by

$$ExpExpish(x) = x \cdot ExpExp(x) = x \cdot e^{-e^{-x}} \tag{14}$$

The derivative of ExpExpish can be calculated by

$$\frac{dExpExpish(x)}{dx} = e^{-e^{-x}}\left(x \cdot e^{-x} + 1\right) \tag{15}$$

where $ExpExp(x) \in (0,1)$ and $ExpExpish(x)$ is a non-monotonic function, which decreases first and then increases from left to right. As seen in Fig. 2d, the curve with the positive half axis is basically coincident with $y = x$, and the minimum value is taken as $x \approx -0.5671$. Thus, it is an unsaturated function with a range of $(-0.0973, +\infty)$. The curve of $\frac{dExpExpish(x)}{dx}$ shows that the derivative is close to 0 in the $(-\infty, -2]$ region, which increases the sparsity of the output. $ExpExp(x)$ is a multiorder derivative smooth function.

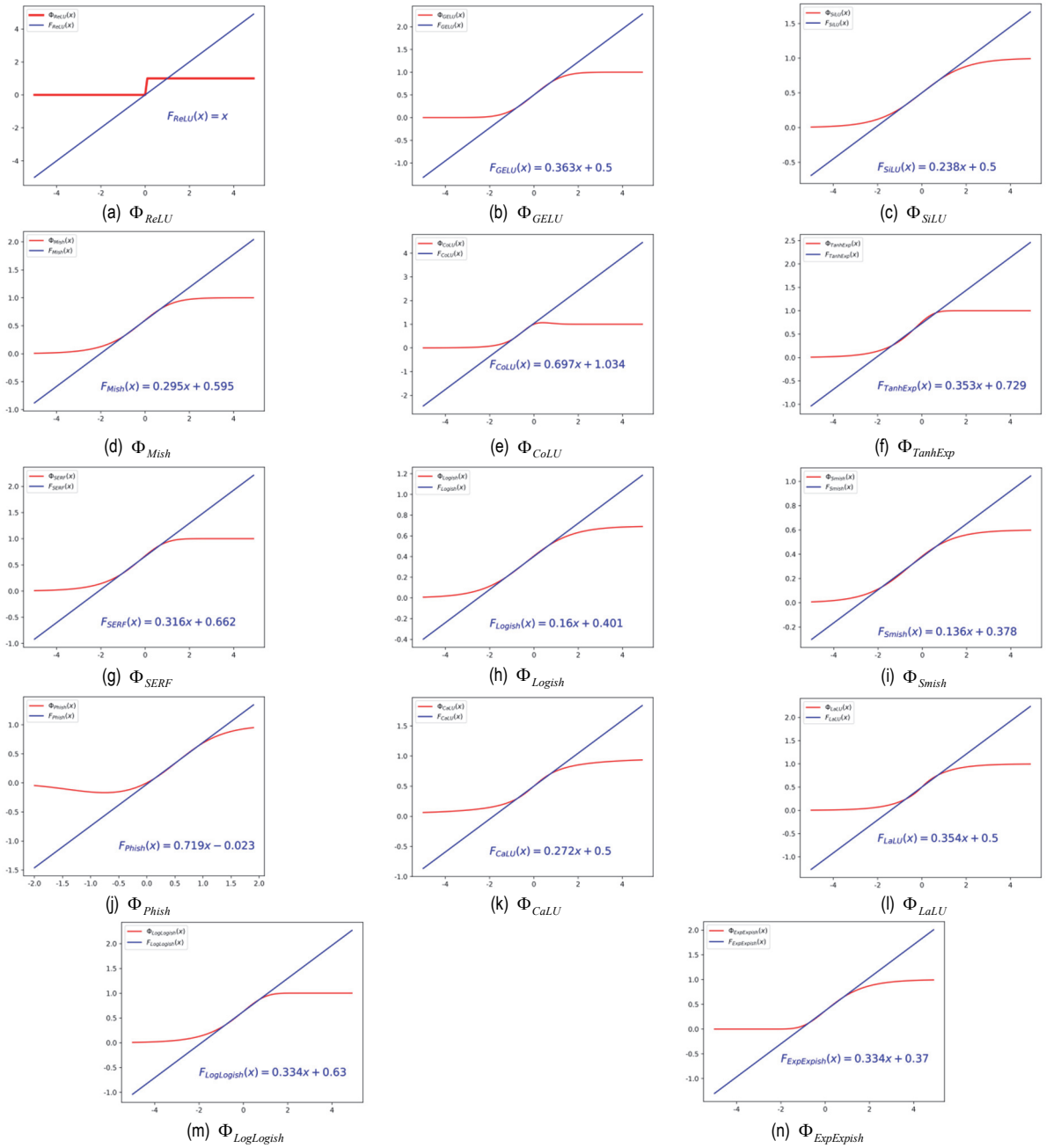
3.2 Analysis of the Activation Function

As shown in Tab. 2, important properties of different AFs are documented. Most of the AFs based on SFs are non-monotonic, and all of them are unsaturated functions. The function values of SFs at $x = 0$ are also different. Fig. 1 and Fig. 2 show that if $\Phi(0)$ is larger, the curve of the AF in the positive region is closer to $f(x) = x$, the training network is more stable, and gradient backpropagation is easier. $\Phi_{CoLU}(0) = 1$ is the largest, and by adjusting α, $\Phi_{ARiA} = 0.5^{\alpha}$ can exceed $\Phi_{CoLU}(0)$. Thus, ARiA can be made closer to $f(x) = x$ by adjusting α.

Table 2 Summary of the properties of multiple activation functions

AFs	Parametric	Monotonic	Smooth	Range	$\Phi(0)$
ReLU	No	Yes	C^0	$[0, \infty)$	1
GELU	No	No	C^∞	$[\approx -0.1636, \infty)$	0.5
SiLU	No	No	C^∞	$[\approx -0.2785, \infty)$	0.5
Swish (β = 1.5)	Yes	No	C^∞	$[\approx -0.156, \infty)$	0.5
E-Swish	Yes	No	C^∞	$[\approx -0.2785\beta, \infty)$	0.5
ARiA (α = 1.5, β = 2)	Yes	No	C^∞	$[\approx -0.1392, \infty)$	$(0.5)^{\alpha}$
CoLU	No	No	C^∞	$[\approx -0.7269, \infty)$	1
Mish	No	No	C^∞	$[\approx -0.3088, \infty)$	0.6
β-Mish	Yes	No	C^∞	$[\approx -0.2650, \infty)$	0.6
TanhExp	No	No	C^∞	$[\approx -0.3532, \infty)$	0.7616
SERF	No	No	C^∞	$[\approx -0.3484, \infty)$	0.6730
Logish	No	No	C^∞	$[\approx -0.2528, \infty)$	0.4055
Smish	No	No	C^∞	$[\approx -0.2500, \infty)$	0.3846
Phish	No	No	C^∞	$[0, \infty)$	0
FTS (S_C Swish}	No	Yes	C^1	$[0, \infty)$	$\Phi(0_-) = 0,$ $\Phi(0_+) = 0.5$
PREU	Yes	No	C^1	$\left[-\alpha/\beta e, \infty\right)$	1
SERLUs	Yes	No	$\begin{cases} C^1, & \alpha = 1 \\ C^0, & \text{otherwise} \end{cases}$	$\left[-\lambda\alpha e^{-1}, \infty\right)$	$\Phi(0_-) = \alpha,$ $\Phi(0_+) = 1$
Hard Swish	No	No	C^0	$[\approx -0.2785, \infty)$	0.5
SCL Mish	No	Yes	C^1	$[0, \infty)$	$\Phi(0_-) = 0,$ $\Phi(0_+) = 0.6$
CaLU	No	Yes	C^∞	$\left[-1/\pi, \infty\right]$	0.5
LaLU	No	No	C^∞	$[-0.1839, \infty]$	0.5
LogLogish	No	No	C^∞	$[-0.3122, \infty]$	0.6321
ExpExpish	No	No	C^∞	$[-0.0973, \infty]$	0.3679

The SF curve shows that each squashing curve near 0 is approximately linear. Therefore, the least squares method is used in [−1, 1], and the SF is fitted with a linear function. The specific fit function is shown in Fig. 3. The figure shows that each SF can be approximated as $f(x)=\alpha x+\beta$ in the range [−1, 1]; however, the fitting parameters of the various SFs differ. The intercept β is 0.5 in the linear fits of Φ_{GELU}, Φ_{SiLU}, Φ_{CaLU} and Φ_{LaLU}, the intercepts of the linear fits of Φ_{Mish}, Φ_{CoLU}, $\Phi_{TanhExp}$, Φ_{SERF} and $\Phi_{LogLogish}$ are greater than 0.5, and the intercepts of the linear fits of the others are all less than 0.5. Moreover, the intercept of the fitting function is essentially the same as $\Phi(0)$. The slopes of each fitted line also differ. When the slope is larger, the function value changes faster, and the corresponding squeezing function saturates faster. Therefore, we use a piecewise function to construct the SFs and then use the piecewise SFs to develop a new AF.



(a) Φ_{ReLU} (b) Φ_{GELU} (c) Φ_{SiLU}
(d) Φ_{Mish} (e) Φ_{CoLU} (f) $\Phi_{TanhExp}$
(g) Φ_{SERF} (h) Φ_{Logish} (i) Φ_{Smish}
(j) Φ_{Phish} (k) Φ_{CaLU} (l) Φ_{LaLU}
(m) $\Phi_{LogLogish}$ (n) $\Phi_{ExpExpish}$

Figure 3 Linear fit curves for different squashing functions

4 AQULU

4.1 Definition

From the above analysis, each SF can be fitted with a linear function in the vicinity of 0, so the overall SF can be fitted with a piecewise function, which can be expressed as:

$$\Phi_{QuLU}(x)=\begin{cases}1 & \text{if } x\geq\dfrac{1-\beta}{\alpha}\\ \alpha x+\beta & \text{if } -\dfrac{\beta}{\alpha}\leq x<\dfrac{1-\beta}{\alpha}\\ 0 & \text{if } x<-\dfrac{\beta}{\alpha}\end{cases} \tag{16}$$

Since there is a linear function in the SF, it is a quadratic function multiplied by x. We define the QuLU as follows:

$$f(x) = x \cdot \Phi_{QuLU}(x) \tag{17}$$

where $0 < \alpha \le 1$ and $\beta \ge 0$ can be either manually defined hyperparameters or training parameters. When $\alpha \to +\infty$, $\beta = 1$, it becomes a ReLU; when $\alpha = \frac{1}{6}$, $\beta = 0.5$, it becomes HardSwish [59]; when $\alpha = 0.5$, $\beta = 1$, it becomes SqREU [61].

If the parameters are trainable, we propose the adaptive quadratic linear unit (AQuLU):

$$\begin{aligned} f(x_i) &= x_i \cdot \Phi_{QuLU}(x_i) \\ &= \begin{cases} x_i & \text{if } x_i \ge \dfrac{1-\beta_i}{\alpha_i} \\ \alpha_i x_i^2 + \beta_i x_i & \text{if } -\dfrac{\beta_i}{\alpha_i} \le x_i < \dfrac{1-\beta_i}{\alpha_i} \\ 0 & \text{if } x_i < -\dfrac{\beta_i}{\alpha_i} \end{cases} \end{aligned} \tag{18}$$

The subscript i indicates that α_i and β_i can vary on different channels. The AQuLU introduces a very small number of extra parameters. The number of extra parameters is equal to 2 times the total number of channels and can be ignored when considering the total number of weights. Thus, there is no extra risk of overfitting.

4.2 Optimization

The AQuLU can be trained using backpropagation and optimized simultaneously with other layers. The update formulations of $\{\alpha_i, \beta_i\}$ are simply derived from the chain rule. The gradient of α_i for one layer is as follows:

$$\frac{\partial \mathcal{E}}{\partial p_i} = \sum_{x_i} \frac{\partial \mathcal{E}}{\partial f(x_i)} \frac{\partial f(x_i)}{\partial p_i} \tag{19}$$

where $\mathcal{E}$ represents the objective function and $p_i \in \{\alpha_i, \beta_i, x_i\}$. The term $\frac{\partial \mathcal{E}}{\partial f(x_i)}$ is the gradient propagated from the deeper layer.

The gradient of α_i is given by

$$\frac{\partial f(x_i)}{\partial \alpha_i} = \begin{cases} x_i^2 & \text{if } -\dfrac{\beta_i}{\alpha_i} \le x_i < \dfrac{1-\beta_i}{\alpha_i} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

The gradient of β_i is given by

$$\frac{\partial f(x_i)}{\partial \beta_i} = \begin{cases} x_i & \text{if } -\dfrac{\beta_i}{\alpha_i} \le x_i < \dfrac{1-\beta_i}{\alpha_i} \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

In this way, the input gradient is

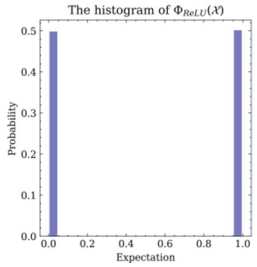
$$\frac{\partial f(x_i)}{\partial x_i} = \begin{cases} 1 & \text{if } x_i \ge \dfrac{1-\beta_i}{\alpha_i} \\ 2\alpha_i x_i + \beta_i & \text{if } -\dfrac{\beta_i}{\alpha_i} \le x_i < \dfrac{1-\beta_i}{\alpha_i} \\ 0 & \text{if } x_i < -\dfrac{\beta_i}{\alpha_i} \end{cases} \tag{22}$$

Note that weight decay (l_2 regularization) should not be used when updating the parameters. A weight decay tends to push α_i and β_i to 0, which makes the AQuLU problematic.

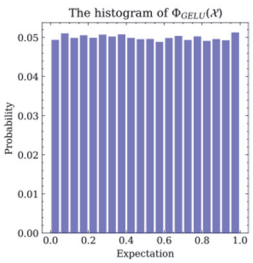
4.3 Initialization

Initialization of parameters α_i and β_i is discussed. Usually, in the commonly used CNN models, a batch normalization layer exists to ensure that the input $(\mathcal{X})$ of each layer satisfies the standard Gaussian distribution. As shown in Theorems 1 and 2, the SFs of GELU, SiLU, Mish, TanhExp, SERF, Logish, Phish, CaLU, LaLU, LogLogish and ExpExpish can all be used as the cumulative distribution function of the variable $\mathcal{X}$. As shown in Theorem 3, if $\mathcal{X}$ is a continuous random variable, the values of the activation functions conform to a uniform distribution of (0, 1).

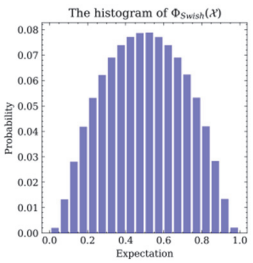
However, $\mathcal{X}$ is a discrete random variable, and its histogram can be used for predictions. Fig. 4 shows the histograms of values for different SFs. The histogram of the GELU squashing function conforms to a uniform distribution, while the Swish squashing functions conform to a normal distribution, and the rest of the squashing functions do not conform to a common distribution. However, the overall distribution is similar to a uniform distribution.



(a) Φ_{ReLU}



(b) Φ_{GELU}



(c) Φ_{SiLU}

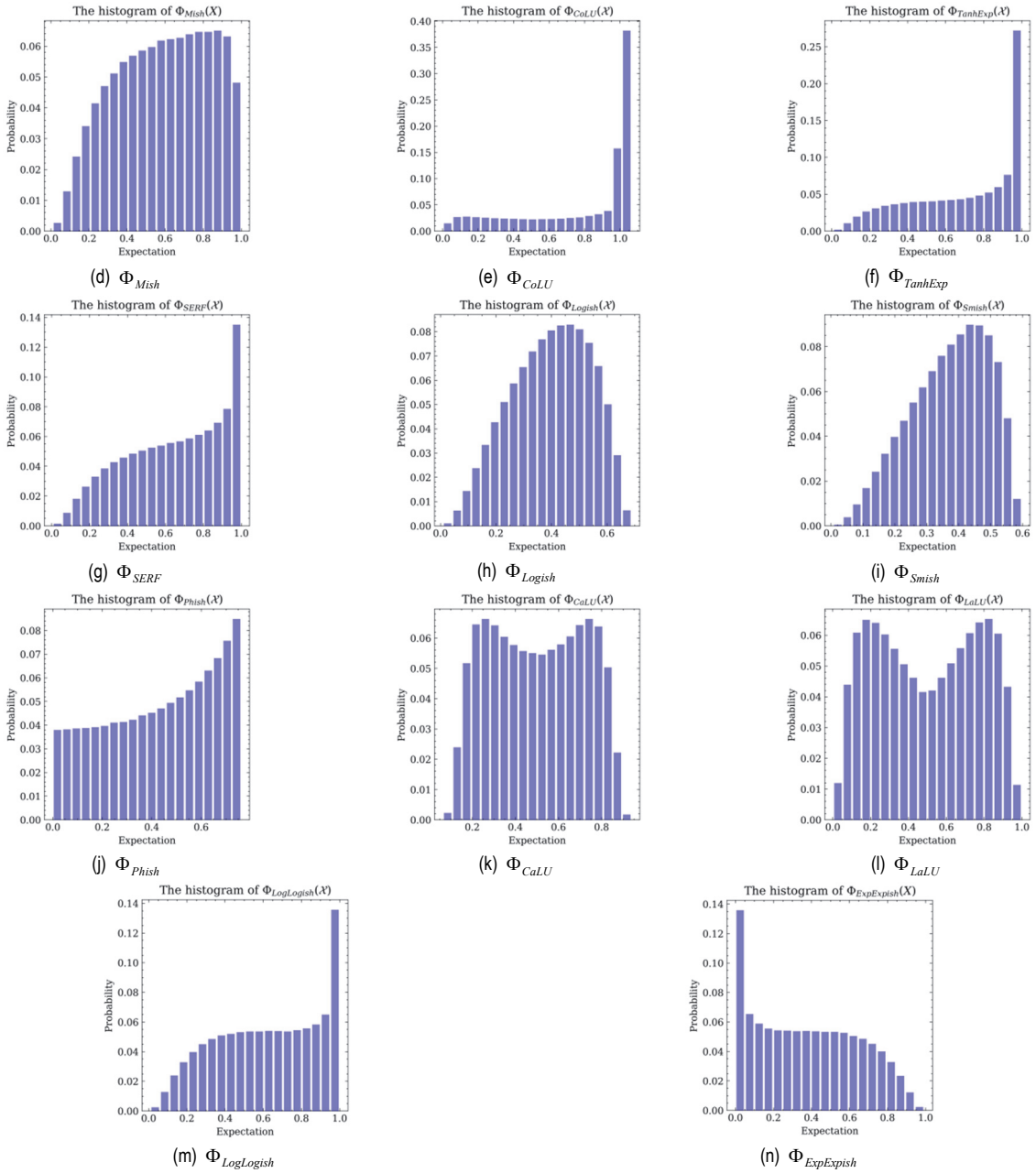


Figure 4 Histograms of different squashing functions when *X* is a standard normal distribution

Theorem 1: Any random variable $\mathcal{X}$ (discrete or continuous) has a distribution function.

Theorem 2: The cumulative distribution function $F_{\mathcal{X}}(x)$ of a random variable $\mathcal{X}$ has three important properties:

- The cumulative distribution function $F_{\mathcal{X}}(x)$ is a nondecreasing function. This follows directly from the previous result we derived: For $a < b$,

$$Pr(a < \mathcal{X} < b) \geq 0 \quad \Rightarrow F_{\mathcal{X}}(b) - F_{\mathcal{X}}(a) \geq 0 \\ \Rightarrow F_{\mathcal{X}}(a) \leq F_{\mathcal{X}}(b) \tag{23}$$

- As $x \to -\infty$, the value of $F_{\mathcal{X}}(x)$ approaches 0 (or is equal to 0). That is, $\lim_{x\to-\infty} F_{\mathcal{X}}(x) = 0$. This follows in part because $Pr(\phi) = 0$.

- As $x \to +\infty$, the value of $F_{\mathcal{X}}(x)$ approaches 1 (or is equal to 1). That is, $\lim_{x\to+\infty} F_{\mathcal{X}}(x) = 0$. This follows in part because $Pr(\varepsilon) = 0$.

Theorem 3: There is a continuous random variable $\mathcal{X}$, and the cumulative distribution function $\mathcal{Y} = F_{\mathcal{X}}(x)$ of *X* is also a continuous function. Then, the value of $\mathcal{Y}$ is also a random variable, and $\mathcal{Y} \sim U(0,1)$.

However, the histograms of Φ_{ReLU}, Φ_{GELU}, Φ_{SiLU}, Φ_{CaLU} and Φ_{LaLU} are symmetric, indicating the same regularization for positive and negative $\mathcal{X}$. Because

$Pr(\Phi_{Mish}(\mathcal{X}) \leq 0.5) < Pr(0.5 < \Phi_{Mish}(\mathcal{X}) \leq 1)$,

$Pr(\Phi_{CoLU}(\mathcal{X}) \leq 0.5) << Pr(0.5 < \Phi_{CoLU}(\mathcal{X}) \leq 1)$,

$Pr(\Phi_{TanhExp}(\mathcal{X}) \leq 0.5) < Pr(0.5 < \Phi_{TanhExp}(\mathcal{X}) \leq 1)$,

$Pr\left(\Phi_{SERF}\left(\mathcal{X}\right)\le 0.5\right) < Pr(0.5 < \Phi_{SERF}\left(\mathcal{X}\right)\le 1)$, and $Pr\left(\Phi_{LogLogish}\left(\mathcal{X}\right)\le 0.5\right) < Pr(0.5 < \Phi_{LogLogish}\left(\mathcal{X}\right)\le 1)$, Mish, CoLU, TanhExp, SERF and LogLogish regularize more when $\mathcal{X}$ is positive. Conversely, Logish, Smish, Phish and ExpExpish regularize more when $\mathcal{X}$ is negative.

We hope that the activation function is approximately equal to the identity function and that the probabilities of ReLU, GELU, SiLU, CaLU and LaLU being less than $0.5X$ and being greater than $0.5X$ are equal. Mish, CoLU, TanhExp, SERF and LogLogish are more likely to be greater than $0.5X$, while Logish, Smish, Phish and ExpExpish are less than $0.5X$. Therefore, the performances of Mish, CoLU, TanhExp, SERF and LogLogish are better. To improve the performance of QuLU, as shown in Fig. 5, $\Phi_{QuLU}\left(x\right)$ must satisfy the following:

$$Pr\left(f\left(\mathcal{X}\right)\le\beta\right) < Pr(\beta < f\left(\mathcal{X}\right)\le 1) \Rightarrow \frac{\beta^2}{2\alpha} < \frac{1-\beta^2}{2\alpha} \tag{24}$$

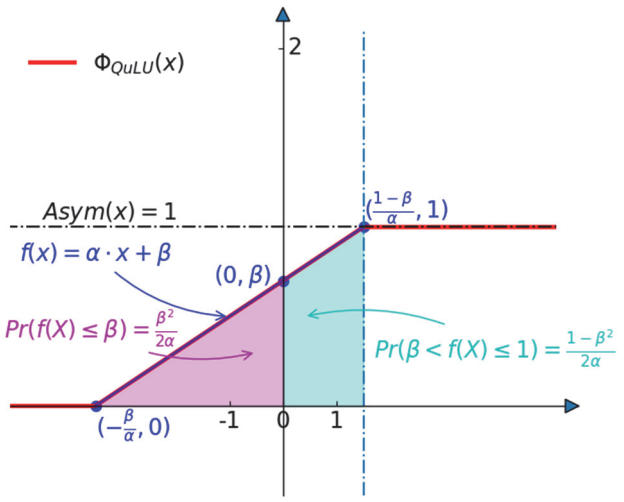


Figure 5 Φ(*x*) of QuLU

Fig. 5 shows the curve of the piecewise squashing functions. For the selection of parameters, α and β are constrained. First, it is necessary to ensure that the [−1, 1] area must be included in the domain of $f\left(x\right)=\alpha x+\beta$. Second, to follow the "3σ ReLU" (in the empirical sciences, the so-called three-sigma rule of thumb expresses a conventional heuristic in which nearly all values are taken to lie within three standard deviations of the mean) of the normal distribution, there are the following constraints:

$$\begin{cases} 1-\beta/\alpha & \le 3 \\ 1-\beta/\alpha & \ge 1 \\ -\beta/\alpha & \le -1 \\ -\beta/\alpha & \ge -3 \\ \alpha & > 0 \\ \beta & > 0 \end{cases} \tag{25}$$

Simultaneously, Eq. (24) and Eq. (25) are used to obtain the solution set for α and β, as shown in Fig. 6.

When the input random variable $\mathcal{X}\sim N\left(0,1\right)$, the squashing function is $F\left(\mathcal{X}\right)=\alpha_i x+\beta_i \sim N\left(\beta,\alpha^2\right)$, where β_i is the mean of $F\left(\mathcal{X}\right)$ and α_i^2 is the variance. The smaller the value of α_i^2 is, the more concentrated the random variable in the mean range. When β_i is close to 1, $x\cdot F\left(x\right)$ is closer to x, i.e., the activation function is closer to the identity function. In general, when α_i is small and β_i is close to 1, AQuLU is closer to the identity function, so the training is more stable and the backpropagation of the gradient is easier. In terms of the gradient, if β_i is closer to 1, α_i is closer to 0 and $2\alpha_i x+\beta_i$ is closer to 1, so the training is not prone to gradient disappearance and explosion. To prevent parameter α_i from updating to zero, α_i is initialized to 7/30, and β_i is initialized to $\sqrt{1/2}$.

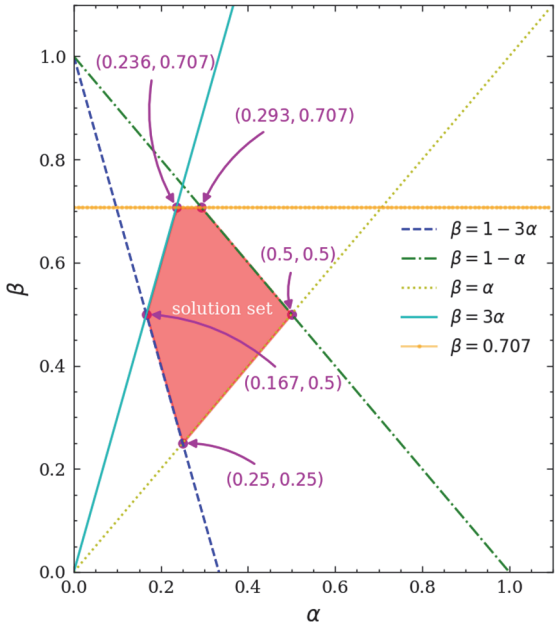


Figure 6 Solution set for α and β

5 EXPERIMENTS

In this section, we further demonstrate the superiority of AQuLU in practice. First we tested the applicability of AQuLU by ablation test. AQuLU is then applied to different advanced frameworks to test their performance. We also present results from ablation studies performed on the MNIST [90] (greyscale images with 10 classes, 60k training examples and 10k test examples) and CIFAR [91] (colour images with 10/100 classes, 50k training examples and 10k test examples) datasets. Overall, our proposed AQuLUs is due to other baseline functions.

5.1 Ablation Studies

The neural network is a complex model. The attributes that affect its training and optimization performance include many hyper-parameters, which further affect the

generalization performance of the network. These hyperparameters include network depths, network widths, learning rates, and optimizers. Here, we analyse and compare the impacts of different hyper-parameters on the selected network through the different AFs, namely, the ReLU, GELU, SiLU, Mish, CoLU, TanhExp, SERF, Logish, Smish, Phish, CaLU, LaLU, LogLogish, ExpExpish, QuLU ($\alpha = \frac{7}{30}, \beta = \sqrt{\frac{1}{2}}$) and AQuLU AFs. To do this, we used the MNIST and CIFAR-10 datasets.

5.1.1 MNIST

Number of layers: The depth of the neural network increases, which makes the model more complex and the process of training and optimization more difficult. In this experiment, the number of Dense Units is 256, and each dense layer is followed by a batch normalization layer. The performance degradation of a variety of different AFs is consistent with the above facts. QuLU, AQuLU, TanhExp, LogLogish, and SERF have higher accuracy than other AFs (Fig. 7a). This makes QuLU and AQuLU suitable for large and complex networks.

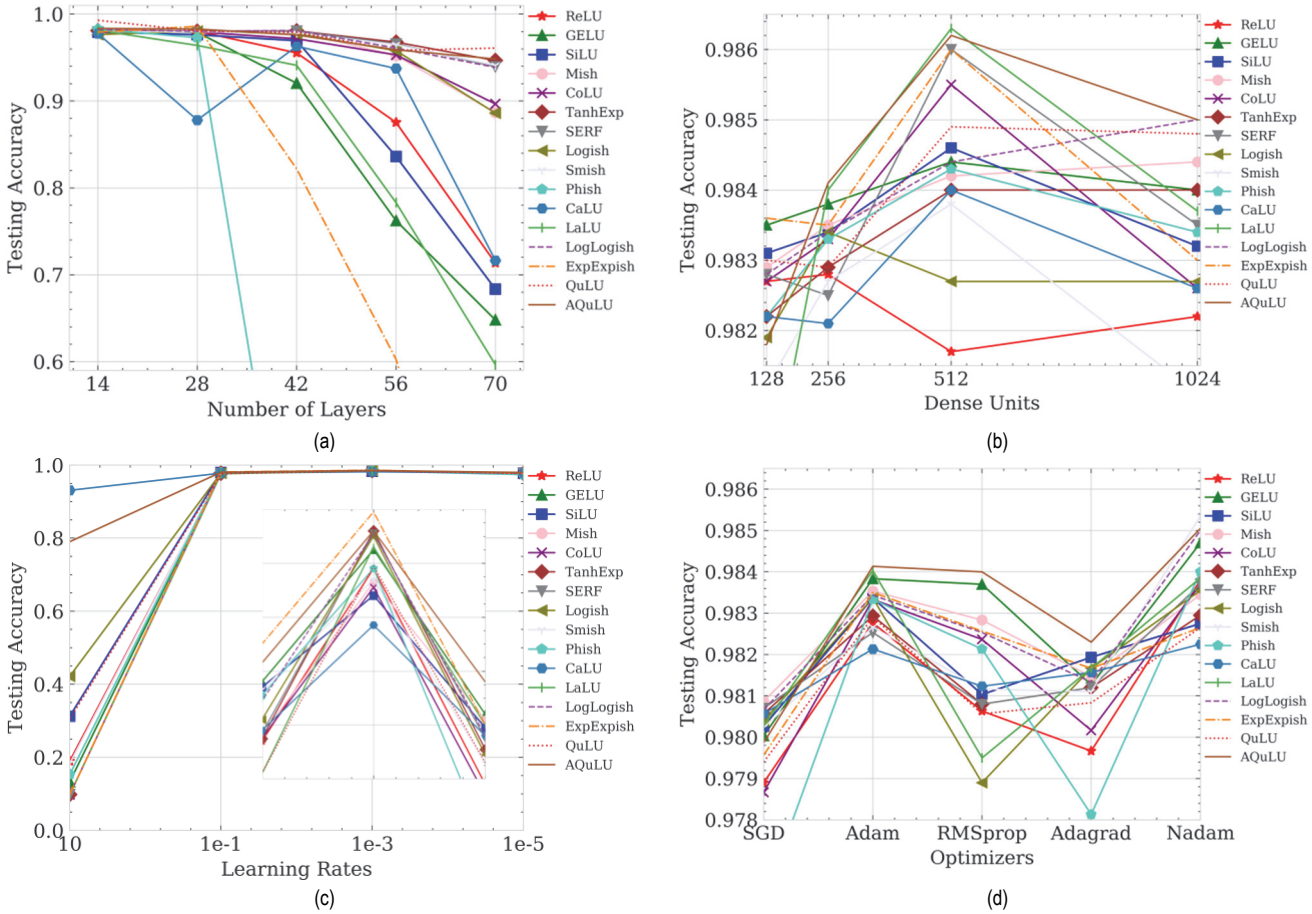


Figure 7 Ablation studies for the MNIST dataset. Top: Testing accuracies vs number of layers 7(a) and dense units 7(b). Bottom: Testing accuracies vs learning rates 7(c) and optimizer 7(d)

Dense Units: In this trial, we used Adam [92] as an optimizer, and the model had 4 layers of depth, each with dense layers and batch normalization layer. The complexity of the model increases as the number of dense units increases, and AQuLU outperforms the other AFs (Fig. 7b). This indicates that AQuLU performs well in complex models.

Learning Rates: At various learning rates, AQuLU performs better than the other activation functions (Fig. 7c). Particularly, with smaller learning rates, the performance of AQuLU does not degrade significantly compared to that of other AFs. This shows that AQuLU is more suitable for complex networks.

Optimizers: In this case, with various optimizers, the overall performance of AQuLU is better than that of the other AFs (Fig. 7d). In the case of the NAdam optimizer, the performance of all activation functions is improved.

5.1.2 CIFAR-10

We use the VGG model with smaller parameters as the feature extraction model of the classification model, and the specific network architecture is shown in Tab. 3. Each activation function model was performed for 50 epochs and run 3 times to take the average.

Number of layers: The performance of all competing activation functions decreases as the number of convolutional layers continues to increase (Fig. 8a); however, the QuLU and AQuLU models maintain the best performance. Adam is used as the optimizer.

Batch Size: As seen in Fig. 8b, the accuracy increases first and then decreases as the batch size increases; however, AQuLU and QuLU occupy a better position in all batch sizes.

Learning Rates: In Fig. 8c, the performance of the model does not perform well for either smaller or larger

learning rates. However, among the many activation functions, AQuLU and QuLU perform the best. Particularly, with smaller learning rates, the performance of AQuLU does not degrade significantly compared to that of other AFs. This shows that AQuLU is more suitable for complex networks.

Table 3 Architecture of convolutional neural networks with different depths

ConvNet Configuration					
A	B	C	D	E	F
11 weight layers	19 weight layers	27 weight layers	35 weight layers	43 weight layers	51 weight layers
input(32 × 32 RGB image)					
conv3-16 × 1	conv3-16 × 2	conv3-16 × 3	conv3-16 × 4	conv3-16 × 5	conv3-16 × 6
maxpool2-2					
conv3-32 × 1	conv3-32 × 2	conv3-32 × 3	conv3-32 × 4	conv3-32 × 5	conv3-32 × 6
maxpool2-2					
conv3-64 × 2	conv3-64 × 4	conv3-64 × 6	conv3-64 × 8	conv3-64 × 10	conv3-64 × 12
maxpool2-2					
conv3-128 × 2	conv3-128 × 4	conv3-128 × 6	conv3-128 × 8	conv3-128 × 10	conv3-128 × 12
maxpool2-2					
conv3-128 × 2	conv3-128 × 4	conv3-128 × 6	conv3-128 × 8	conv3-128 × 10	conv3-128 × 12
maxpool2-2					
FC-128					
FC-128					
FC-10					
SOFT-MAX					

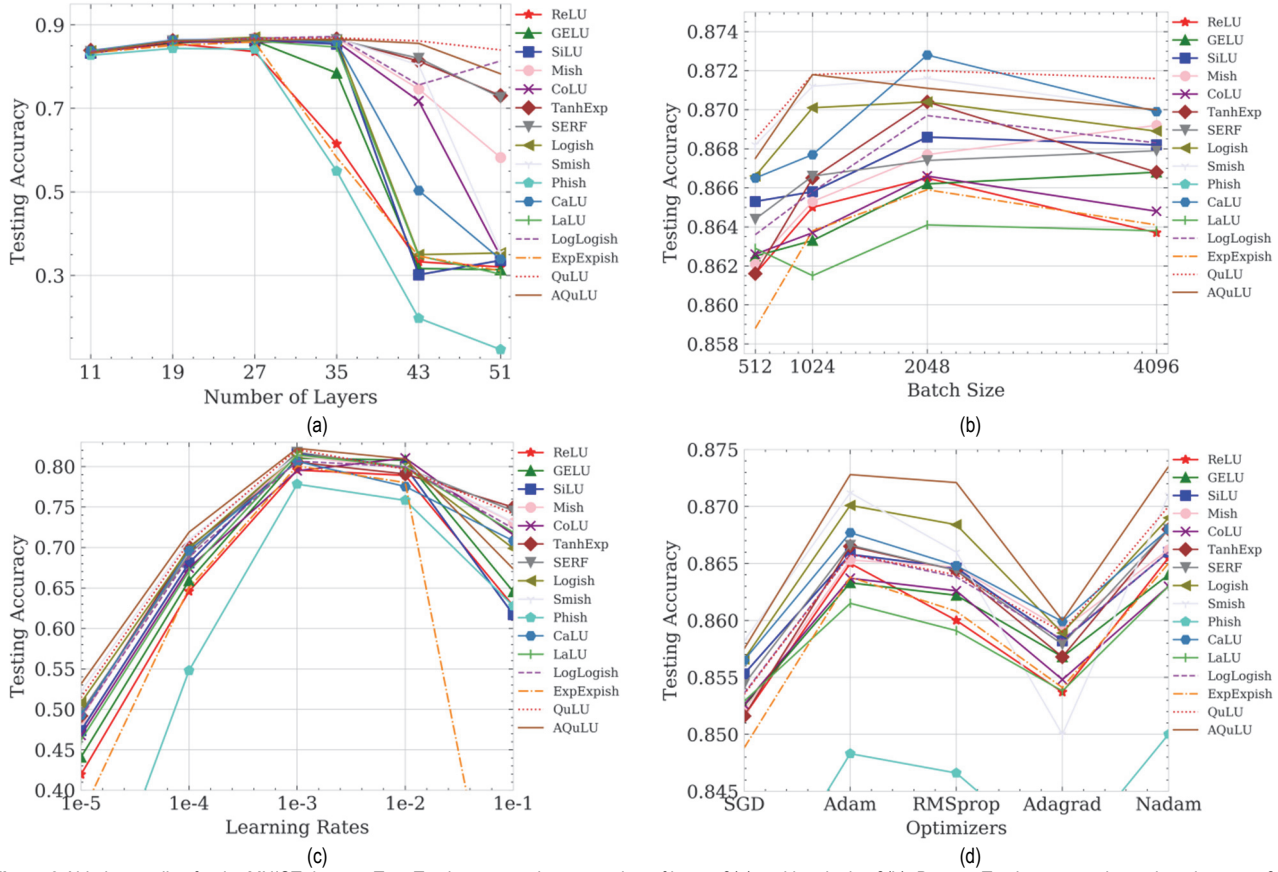


Figure 8 Ablation studies for the MNIST dataset. Top: Testing accuracies vs number of layers 8(a) and batch size 8(b). Bottom: Testing accuracies vs learning rates 8(c) and optimizer 8(d)

Optimizers: In this case, the overall performance of AQuLU is equal to or slightly better than that of other AFs using different optimizers (Fig. 8d). In the case of AdaGrad and the SGD optimizer, a decrease in performance can be noticed for all AFs.

5.2 Image Classification

For image classification, we use different state-of-the-art frameworks in CIFAR10 and CIFAR100 to judge performance.

5.2.1 CIFAR-10

We use a variety of standard and advanced neural network structures to change the activation function under the premise of other hyper-parameters unchanged, and compare the performance of our designed activation function with other baseline functions. Among the baseline activation functions, ReLU, Mish, SERF, TanhExp and LogLogish are chosen because these AFs can be adapted to larger and more complex models. We compare the baseline activation, QuLU and AQuLU functions on a variety of

architectures designed for CIFAR10: SqueezeNet [93], ResNet-50 [94], GoogLeNet [95], DenseNet121 [96], DenseNet169 [96], ResNeXt29_32×4d [97], MobileNet-v2 [98], and ShuffleNet-v2 [99]. The initial learning rate is, and it is adjusted by decay. The optimizer is Adam. Each model has a batch size of 128 and runs 5 times, and 50 epochs are performed.

Tab. 4 shows that AQuLU mostly outperforms the baseline activation function for all standard architectures used in the experiments. LogLogish performs best in MobileNet-v2. QuLU also performs well, outperforming most of the baseline activation functions. AQuLU improved 36.76% over the baseline ReLU in the SqueezeNet model, typically providing a 0.4% to 3% performance improvement over other network architectures enabled by the baseline ReLU.

5.2.2 CIFAR-10

Here, we turn to another dataset, CIFAR-100, which is similar to CIFAR-10 in that it has 100 classes. The specific experimental details and models are consistent with those of CIFAR-10, and the results are shown in Tab. 5. From the table, we can conclude that AQuLU performs better than the baseline activation function for most models, for example, 10.84% better than ReLU, 2% better than Mish, and 2.23% better than SERF on the SqueezeNet model. The results show that our proposed activation function is stable and effective on challenging datasets.

Table 4 Compare test accuracy based on CIFAR-10 image classification in various network architectures

Methods	ReLU	Mish	SERF	TanhExp	LogLogish	QuLU (Ours)	AQuLU (Ours)
SqueezeNet	59.47%	75.67%	76.77%	72.71%	74.72%	80.58%	81.33%
ResNet-50	91.71%	90.99%	91.93%	91.32%	91.41%	92.09%	92.24%
GoogLeNet	90.98%	91.15%	91.26%	91.18%	91.25%	91.25%	91.39%
DenseNet121	89.09%	91.46%	91.67%	91.26%	91.82%	91.88%	92.34%
DenseNet169	90.80%	91.82%	91.98%	91.36%	91.86%	92.03%	92.53%
ResNeXt29 32×4d	91.59%	92.27%	92.28%	92.13%	92.23%	92.13%	92.49%
MobileNet-v2	87.51%	88.54%	88.70%	88.90%	89.47%	88.62%	89.42%
ShuffleNet-v2	77.56%	81.57%	81.65%	81.62%	81.91%	81.83%	82.07%

Table 5 Compare test accuracy based on CIFAR-100 image classification in various network architectures

Methods	ReLU	Mish	SERF	TanhExp	LogLogish	QuLU (Ours)	AQuLU (Ours)
squeezeNet	41.39%	44.98%	44.88%	45.78%	45.65%	44.95%	45.88%
ResNet50	65.88%	68.70%	69.07%	68.80%	68.32%	69.41%	69.21%
GoogLeNet	70.52%	70.19%	69.93%	70.31%	70.72%	70.21%	70.55%
DensenNet121	70.07%	70.55%	69.99%	70.29%	69.95%	70.20%	70.69%
DenseNet169	69.73%	68.66%	69.76%	70.61%	69.28%	70.13%	70.71%
ResNeXt29 32×4d	70.81%	71.74%	71.58%	70.56%	71.34%	71.76%	71.88%
MobileNet-v2	57.37%	57.77%	57.06%	56.50%	57.02%	57.48%	57.87%
ShuffleNet-v2	59.79%	61.31%	61.23%	61.04%	61.32%	60.66%	61.57%

6 CONCLUSION

In this paper, four activation functions are proposed according to GELU et al., namely, CaLU, LaLU, LogLogish and ExpExpish. Analysis shows that every squashing function can be fitted by a linear function $f(x)=\alpha x+\beta$ near 0. The pros and cons of AFs are hidden in the fitting linear function when the input $X\sim N(0,1)$, $f(X)=\alpha x+\beta\sim N(\beta,\alpha^2)$. When β is larger and α is smaller, the activation function performs better. Therefore, we propose a new adaptive segmental activation function, the adaptive quadratic linear unit, abbreviated AQuLU, for more complex, large neural networks. The equation for AQuLU is

$$f(x_i)=\begin{cases} x_i & \text{if } x_i \geq \dfrac{1-\beta_i}{\alpha_i} \\ \alpha_i x_i^2+\beta_i x_i & \text{if } -\dfrac{\beta_i}{\alpha_i}\leq x_i < \dfrac{1-\beta_i}{\alpha_i} \\ 0 & \text{if } x_i < -\dfrac{\beta_i}{\alpha_i} \end{cases} \tag{26}$$

It has properties such as upper unboundedness, lower boundedness, and nonmonotonicity, which are the desired properties for an AF. Flexibility is provided for the function shape by two parameters α and β so that it can be tuned to a specific neural network-based task by adaptively adjusting the parameters. In addition, since QuLU and AQuLU do not involve any exponential, logarithmic, or power operations, they perform computations faster relative to other nonmonotonic, smooth activation functions. Moreover, we propose for the first time that the adaptive quadratic linear unit is used as the activation function, which is the most prominent in the deeper network and can well resist the disappearance of the gradient. We propose ideas and methods to construct an activation function for researchers.

In the ablation test, AQuLU showed strong adaptability and worked well with each hyper-parameter of the neural network. AQuLU also performs more prominently in image classification, where it performs particularly well in weight-based neural networks due to its adaptive nature and large gradients, while its performance in lightweight neural networks is comparable to that of other activation functions. Overall, AQuLU is a simple, effective and versatile activation function that can be incorporated into any neural network for better training and performance gains. Future work will focus on using AQuLU in other computer vision tasks, such as object detection, object segmentation, etc., to explore newer, larger models, and to compare with other adaptive activation functions. And put forward a more unified and recognized comparison scheme.

Acknowledgement

This research was funded by the National Natural Science Foundation of China (Grant Numbers 32171913 and 32001418).

7 REFERENCES

[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT press. https://doi.org/10.4258/hir.2016.22.4.351

[2] Lederer, J. (2021). Activation functions in artificial neural networks: A systematic overview.

[3] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533-536. https://doi.org/10.1038/323533a0

[4] Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of Machine Learning Research*.

[5] Glorot, X., Bordes, A., & Bengio, Y. (2011a). Deep sparse rectifier neural networks. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315-323.

[6] Dahl, G. E., Sainath, T. N., & Hinton, G. E. (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8609-8613. https://doi.org/10.1109/ICASSP.2013.6639346

[7] Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus).

[8] Lu, L. (2020). Dying ReLU and Initialization: Theory and Numerical Examples. *Communications in Computational Physics*, *28*(5), 1671-1706. https://doi.org/10.4208/cicp.oa-2020-0165

[9] Maas, A. L., Hannun, A. Y., Ng, A. Y. et al. (2013). Rectifier nonlinearities improve neural network acoustic models. *Proceedings of Machine Learning Research*, *30*(1), 3.

[10] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE International Conference on Computer Vision*, 1026-1034. https://doi.org/10.1109/ICCV.2015.123

[11] Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. ArXiv Preprint ArXiv: 1505.00853.

[12] Shang, W., Sohn, K., Almeida, D., & Lee, H. (2016). Understanding and improving convolutional neural networks via concatenated rectified linear units. *International Conference on Machine Learning*, 2217-2225.

[13] Cao, J., Pang, Y., Li, X., & Liang, J. (2018). Randomly translational activation inspired by the input distributions of ReLU. *Neurocomputing*, *275*, 859-868. https://doi.org/10.1016/j.neucom.2017.09.031

[14] Macêdo, D., Zanchettin, C., Oliveira, A. L., & Ludermir, T. (2019). Enhancing batch normalized convolutional networks using displaced rectifier linear units: A systematic comparative study. *Expert Systems with Applications*, *124*, 271-281. https://doi.org/10.1016/j.eswa.2019.01.066

[15] Carlile, B., Delamarter, G., Kinney, P., Marti, A., & Whitney, B. (2017). Improving deep learning by inverse square root linear units (ISRLUs). ArXiv Preprint ArXiv: 1710.09967.

[16] Lin, G. & Shen, W. (2018). Research on convolutional neural network based on improved Relu piecewise activation function. *Procedia Computer Science*, *131*, 977-984. https://doi.org/10.1016/j.procs.2018.04.239

[17] Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. *Advances in Neural Information Processing Systems*, 30.

[18] Trottier, L., Giguere, P., & Chaib-Draa, B. (2017). Parametric exponential linear unit for deep convolutional neural networks. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 207-214. https://doi.org/10.1109/ICMLA.2017.00038

[19] Barron, J. T. (2017). Continuously differentiable exponential linear units. ArXiv Preprint ArXiv: 1704.07483.

[20] Li, Y., Fan, C., Li, Y., Wu, Q., & Ming, Y. (2018). Improving deep neural network with multiple parametric exponential linear units. *Neurocomputing*, *301*, 11-24. https://doi.org/10.1016/j.neucom.2018.01.084

[21] Qiumei, Z., Dan, T., & Fenghua, W. (2019). Improved convolutional neural network based on fast exponentially linear unit activation function. *IEEE Access*, *7*, 151359-151367. https://doi.org/10.1109/ACCESS.2019.2948112

[22] Kim, D., Kim, J., & Kim, J. (2020). Elastic exponential linear units for convolutional neural networks. *Neurocomputing*, *406*, 253-266. https://doi.org/10.1016/j.neucom.2020.03.051

[23] Cheng, Q., Li, H., Wu, Q., Ma, L., & Ngan, K. N. (2020). Parametric deformable exponential linear units for deep neural networks. *Neural Networks*, *125*, 281-289. https://doi.org/10.1016/j.neunet.2020.02.012

[24] Krizhevsky, A. & Hinton, G. (2010). Convolutional deep belief networks on cifar-10. *Unpublished Manuscript*, *40*(7), 1-9.

[25] Liew, S. S., Khalil-Hani, M., & Bakhteri, R. (2016). Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems. *Neurocomputing*, *216*, 718-734. https://doi.org/10.1016/j.neucom.2016.08.037

[26] Jiang, X., Pang, Y., Li, X., Pan, J., & Xie, Y. (2018). Deep neural networks with elastic rectified linear units for object recognition. *Neurocomputing*, *275*, 1132-1139. https://doi.org/10.1016/j.neucom.2017.09.056

[27] Liu, Y., Zhang, J., Gao, C., Qu, J., & Ji, L. (2019). Natural-logarithm-rectified activation function in convolutional neural networks. *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, 2000-2008. https://doi.org/10.1109/ICCC47050.2019.9064398

[28] Dubey, S. R. & Chakraborty, S. (2021). Average biased ReLU based CNN descriptor for improved face retrieval. *Multimedia Tools and Applications*, *80*(15), 23181-23206. https://doi.org/10.1007/s11042-020-10269-x

[29] Li, B., Tang, S., & Yu, H. (2019). PowerNet: Efficient representations of polynomials and smooth functions by deep neural networks with rectified power units. ArXiv Preprint ArXiv: 1909.05136.

[30] Mercioni, M. A. & Holban, S. (2021). Soft-Clipping Swish: A Novel Activation Function for Deep Learning. *2021 IEEE 15th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 225-230. https://doi.org/10.1109/SACI51354.2021.9465622

[31] Holban, M. A. M. (2021). Soft Clipping Mish - A Novel Activation Function for Deep Learning. *2021 4th International Conference on Information and Computer Technologies (ICICT)*.

[32] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958.

[33] Krueger, D., Maharaj, T., Kramár, J., Pezeshki, M., Ballas, N., Ke, N. R., Goyal, A., Bengio, Y., Courville, A., & Pal, C. (2016). Zoneout: Regularizing rnns by randomly preserving hidden activations. ArXiv Preprint ArXiv: 1606.01305.

[34] Hendrycks, D. & Gimpel, K. (2016). Gaussian error linear units (gelus). ArXiv Preprint ArXiv:1606.08415.

[35] Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. ArXiv Preprint ArXiv:1710.05941.

[36] Misra, D. (2019). Mish: A self regularized non-monotonic neural activation function. ArXiv Preprint ArXiv: 1908.08681, *4*(2), 10-48550.
[37] Liu, X. & Di, X. (2021). TanhExp: A smooth activation function with high convergence speed for lightweight neural networks. *IET Computer Vision*, *15*(2), 136-150. https://doi.org/10.1049/cvi2.12020
[38] Nag, S. & Bhattacharyya, M. (2021). SERF: Towards better training of deep neural networks using log-Softplus ERror activation function. ArXiv Preprint ArXiv: 2108.09598.
[39] Zhu, H., Zeng, H., Liu, J., & Zhang, X. (2021). Logish: A new nonlinear nonmonotonic activation function for convolutional neural network. *Neurocomputing*, *458*, 490-499. https://doi.org/10.1016/j.neucom.2021.06.067
[40] Wang, X., Ren, H., & Wang, A. (2022). Smish: A Novel Activation Function for Deep Learning Methods. *Electronics*, *11*(4), 540. https://doi.org/10.3390/electronics11040540
[41] Patwardhan, N., Ingalhalikar, M., & Walambe, R. (2018). ARiA: Utilizing Richard's Curve for Controlling the Non-monotonicity of the Activation Function in Deep Neural Nets. ArXiv Preprint ArXiv: 1805.08878.
[42] Alcaide, E. (2018). E-swish: Adjusting activations to different network depths. ArXiv Preprint ArXiv: 1801.07145.
[43] Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. *International Conference on Machine Learning*, 1319-1327.
[44] Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2014). Learning activation functions to improve deep neural networks. ArXiv Preprint ArXiv: 1412.6830.
[45] Godfrey, L. B. (2019). An evaluation of parametric activation functions for deep learning. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 3006-3011. https://doi.org/10.1109/SMC.2019.8913972
[46] Wang, X., Qin, Y., Wang, Y., Xiang, S., & Chen, H. (2019). ReLTanh: An activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis. *Neurocomputing*, *363*, 88-98. https://doi.org/10.1016/j.neucom.2019.07.017
[47] Tavakoli, M., Agostinelli, F., & Baldi, P. (2021). Splash: Learnable activation functions for improving accuracy and adversarial robustness. *Neural Networks*, *140*, 1-12. https://doi.org/10.1016/j.neunet.2021.02.023
[48] Li, H., Ouyang, W., & Wang, X. (2016). Multi-bias non-linear activation in deep neural networks. *International Conference on Machine Learning*, 221-229.
[49] Gu, S., Li, W., Gool, L. V., & Timofte, R. (2019). Fast image restoration with multi-bin trainable linear units. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4190-4199. https://doi.org/10.1109/ICCV.2019.00429
[50] Scardapane, S., Van Vaerenbergh, S., Totaro, S., & Uncini, A. (2019). Kafnets: Kernel-based non-parametric activation functions for neural networks. *Neural Networks*, *110*, 19-32. https://doi.org/10.1016/j.neunet.2018.11.002
[51] Scardapane, S., Scarpiniti, M., Comminiello, D., & Uncini, A. (2017). Learning activation functions from data using cubic spline interpolation. *Italian Workshop on Neural Nets*, 73-83. https://doi.org/10.1007/978-3-319-95098-3_7
[52] Molina, A., Schramowski, P., & Kersting, K. (2019). Pad\backslash'e Activation Units: End-to-end Learning of Flexible Activation Functions in Deep Networks. ArXiv Preprint ArXiv: 1907.06732.
[53] Ma, N., Zhang, X., Liu, M., & Sun, J. (2021). Activate or not: Learning customized activation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8032-8042. https://doi.org/10.1109/CVPR46437.2021.00794
[54] Sütfeld, L. R., Brieger, F., Finger, H., Füllhase, S., & Pipa, G. (2020). Adaptive blending units: *Trainable activation functions for deep neural networks. Science and Information Conference*, 37-50. https://doi.org/10.1007/978-3-030-52243-8_4
[55] Asif, A. & others. (2019). Learning Neural Activations. ArXiv Preprint ArXiv: 1912.12187.
[56] Jagtap, A. D., Kawaguchi, K., & Em Karniadakis, G. (2020). Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. *Proceedings of the Royal Society A*, *476*(2239), 20200334. https://doi.org/10.1098/rspa.2020.0334
[57] Jagtap, A. D., Kawaguchi, K., & Karniadakis, G. E. (2020). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, *404*, 109136. https://doi.org/10.1016/j.jcp.2019.109136
[58] Jagtap, A. D., Shin, Y., Kawaguchi, K., & Karniadakis, G. E. (2022). Deep Kronecker neural networks: A general framework for neural networks with adaptive activation functions. *Neurocomputing*, *468*, 165-180. https://doi.org/10.1016/j.neucom.2021.10.036
[59] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., & others. (2019). Searching for mobilenetv3. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1314-1324. https://doi.org/10.1109/ICCV.2019.00140
[60] Avenash, R. & Viswanath, P. (2019). Semantic Segmentation of Satellite Images using a Modified CNN with Hard-Swish Activation Function. *VISIGRAPP (4: VISAPP)*, 413-420. https://doi.org/10.5220/0007469600002108
[61] Wuraola, A., Patel, N., & Nguang, S. K. (2021). Efficient activation functions for embedded inference engines. *Neurocomputing*, *442*, 73-88. https://doi.org/10.1016/j.neucom.2021.02.030
[62] Jin, X., Xu, C., Feng, J., Wei, Y., Xiong, J., & Yan, S. (2016). Deep learning with s-shaped rectified linear activation units. *Proceedings of the AAAI Conference on Artificial Intelligence*, *30*(1). https://doi.org/10.1609/aaai.v30i1.10287
[63] Basirat, M. & Roth, P. (2020). L* ReLU: piece-wise linear activation functions for deep fine-grained visual categorization. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 1218-1227. https://doi.org/10.1109/WACV45572.2020.9093485
[64] Bawa, V. S. & Kumar, V. (2019). Linearized sigmoidal activation: A novel activation function with tractable non-linear characteristics to boost representation capability. *Expert Systems with Applications*, *120*, 346-356. https://doi.org/10.1016/j.eswa.2018.11.042
[65] Nicolae, A. (2018). PLU: The piecewise linear unit activation function. ArXiv Preprint ArXiv: 1809.09534.
[66] Mohit, J. (2018). A New Hyperbolic Tangent Based Activation Function for Neural Networks.
[67] Wilder, R. (2018). SineReLU - An Alternative to the ReLU Activation Function.
[68] Ying, Y., Su, J., Shan, P., Miao, L., Wang, X., & Peng, S. (2019). Rectified exponential units for convolutional neural networks. *IEEE Access*, *7*, 101633-101640. https://doi.org/10.1109/ACCESS.2019.2928442
[69] Yu, C. & Su, Z. (2019). Symmetrical Gaussian Error Linear Units (SGELUs). ArXiv Preprint ArXiv: 1911.03925.
[70] Le Cun, Y. et al. (1989). Generalization and network design strategies. *Connectionism in Perspective*, *19*(143-155), 18.
[71] Elfwing, S., Uchibe, E., & Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, *107*, 3-11. https://doi.org/10.1016/j.neunet.2017.12.012
[72] Vagerwal, A. (2021). Deeper Learning with CoLU Activation. ArXiv Preprint ArXiv: 2112.12078.
[73] Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., & Garcia, R. (2000). Incorporating second-order functional knowledge

for better option pricing. *Advances in Neural Information Processing Systems*, *13*.
[74] Mathayo, P. B. & Kang, D.-K. (2022). Beta and Alpha Regularizers of Mish Activation Functions for Machine Learning Applications in Deep Neural Networks. *International Journal of Internet, Broadcasting and Communication*, *14*(1), 136-141.
[75] Naveen, P. (2022). Phish: A novel hyper-optimizable activation function. https://doi.org/10.36227/techrxiv.17283824.v2
[76] Chieng, H. H., Wahid, N., Ong, P., & Perla, S. R. K. (2018). Flatten-T Swish: A thresholded ReLU-Swish-like activation function for deep learning. ArXiv Preprint ArXiv: 1812.06247. https://doi.org/10.26555/ijain.v4i2.249
[77] Zhang, G. & Li, H. (2018). Effectiveness of scaled exponentially-regularized linear units (SERLUs). ArXiv Preprint ArXiv: 1807.10117.
[78] Apicella, A., Donnarumma, F., Isgrò, F., & Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, *138*, 14-32. https://doi.org/10.1016/j.neunet.2021.01.026
[79] Hu, Z. (1992). The study of neural network adaptive control systems. *Control and Decision*, *7*, 361-366.
[80] Singh, Y. & Chandra, P. (2003). A class+ 1 sigmoidal activation functions for FFANNs. *Journal of Economic Dynamics and Control*, *28*(1), 183-187. https://doi.org/10.1016/S0165-1889(02)00157-4
[81] Qian, S., Liu, H., Liu, C., Wu, S., & Wong, H. S. (2018). Adaptive activation functions in convolutional neural networks. *Neurocomputing*, *272*, 204-212. https://doi.org/10.1016/j.neucom.2017.06.070
[82] Manessi, F. & Rozza, A. (2018). Learning combinations of activation functions. *2018 24th International Conference on Pattern Recognition (ICPR)*, 61-66. https://doi.org/10.1109/ICPR.2018.8545362
[83] Kiliçarslan, S. & Celik, M. (2021). RSigELU: a nonlinear activation function for deep neural networks. *Expert Systems with Applications*, *174*, 114805. https://doi.org/10.1016/j.eswa.2021.114805
[84] Klabjan, D. & Harmon, M. (2019). Activation ensembles for deep neural networks. *2019 IEEE International Conference on Big Data (Big Data)*, 206-214. https://doi.org/10.1109/BigData47090.2019.9006069
[85] Mnih, V., Heess, N., Graves, A. et al. (2014). Recurrent models of visual attention. *Advances in Neural Information Processing Systems*, 2204-2212.
[86] Yin, W., Schütze, H., Xiang, B., & Zhou, B. (2016). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, *4*, 259-272. https://doi.org/10.1162/tacl_a_00097
[87] Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
[88] Gomes, G. S. da S., Ludermir, T. B., & Lima, L. M. (2011). Comparison of new activation functions in neural network for forecasting financial time series. *Neural Computing and Applications*, *20*(3), 417-439. https://doi.org/10.1007/s00521-010-0407-3
[89] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541-551. https://doi.org/10.1162/neco.1989.1.4.541
[90] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. https://doi.org/10.1109/5.726791
[91] Krizhevsky, A., Hinton, G. et al. (2009). Learning multiple layers of features from tiny images.
[92] Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. ArXiv Preprint ArXiv: 1412.6980.
[93] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and < 0.5 MB model size. ArXiv Preprint ArXiv: 1602.07360.
[94] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770-778. https://doi.org/10.1109/CVPR.2016.90
[95] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1-9. https://doi.org/10.1109/CVPR.2015.7298594
[96] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700-4708. https://doi.org/10.1109/CVPR.2017.243
[97] Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1492-1500. https://doi.org/10.1109/CVPR.2017.634
[98] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linearbottlenecks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4510-4520. https://doi.org/10.1109/CVPR.2018.00474
[99] Ma, N., Zhang, X., Zheng, H.-T., & Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient CNN architecture design. *Proceedings of the European Conference on Computer Vision (ECCV)*, 116-131. https://doi.org/10.1007/978-3-030-01264-9_8

Contact information:

Zhandong WU
College of Biological and Agricultural Engineering,
Jilin University,
Changchun, 130022, China

Haiye YU
College of Biological and Agricultural Engineering,
Jilin University,
Changchun, 130022, China

Lei ZHANG
(Corresponding author)
College of Biological and Agricultural Engineering,
Jilin University,
Changchun, 130022, China
E-mail: z_lei@jlu.edu.cn

Yuanyuan SUI
College of Biological and Agricultural Engineering,
Jilin University,
Changchun, 130022, China