IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 August 2023, accepted 29 August 2023, date of publication 14 September 2023, date of current version 21 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3315308

RESEARCH ARTICLE

PolyLU: A Simple and Robust Polynomial-Based Linear Unit Activation Function for Deep Learning

HAN-SHEN FENG[1] AND CHENG-HSIUNG YANG[1,2]
[1]Graduate Institute of Automation and Control, National Taiwan University of Science and Technology, Taipei City 10607, Taiwan
[2]Advanced Manufacturing Research Center, National Taiwan University of Science and Technology, Taipei City 10607, Taiwan

Corresponding author: Cheng-Hsiung Yang (yangch@mail.ntust.edu.tw)

ABSTRACT The activation function has a critical influence on whether a convolutional neural network in deep learning can converge or not; a proper activation function not only makes the convolutional neural network converge faster but also can reduce the complexity of convolutional neural network architecture and gets the same or better performance. Many activation functions have been proposed; however, various activation functions have advantages, defects, and applicable network architectures. A new activation function called Polynomial Linear Unit (PolyLU) is proposed in this paper to improve some of the shortcomings of the existing activation functions. The PolyLU meets the following basic properties: continuously differentiable, approximate identity near the origin, unbounded for positive inputs, bounded for negative inputs, smooth, monotonic, and zero-centered. There is a polynomial term for the negative inputs and no exponential terms in the PolyLU that reduce the computational complexity of the network. Compared to those common activation functions like Sigmoid, Tanh, ReLU, LeakyReLU, ELU, Mish, and Swish, the experiments show that the PolyLU has improved some network complexity and has better accuracy over MNIST, Kaggle Cats and Dogs, CIFAR-10 and CIFAR-100 datasets. Test by the CIFAR-100 dataset with batch normalization, PolyLU improves by 0.62%, 2.82%, 2.44%, 1.33%, 2.08%, and 4.26% of accuracy than ELU, Swish, Mish, Leaky ReLU, ReLU, and Tanh respectively. Test by the CIFAR-100 dataset without batch normalization, PolyLU improves by 1.24%, 4.39%, 2.12%, 5.43%, 15.51%, and 8.10% of accuracy than ELU, Swish, Mish, Leaky ReLU, ReLU, and Tanh respectively.

INDEX TERMS Activation function, convolutional neural network, deep learning, PolyLU.

I. INTRODUCTION

The Universal Approximation theorem for neural networks [1], [2], [3], [4] implies that any continuous function can be approximated by a single-layer neural network with a finite number of neurons; the theorem has been proven with the Sigmoid activation function [5], [6] and was extended to any specific activation function [7], [8]. The General Approximation theorem claims that ''every continuous function can be arbitrarily accurately approximated with a continuous nonlinear function of one variable'' [9]. A sophisticated convolutional neural network architecture in deep learning needs one or more proper activation functions for convergence; a proper activation function not only makes the convolutional neural network converge faster but also can reduce the complexity of a convolutional neural network architecture and gets the same or better performance. There are many activation functions have been proposed; the earliest activation function Sigmoid, Tanh, Softplus [10]; the most popular activation function ReLU [11], and the later LeakyReLU [12], ELU [13], Swish [14], Mish [15]. However, various activation functions have advantages, defects, and suitable application network architectures.

One of the main issues for convolutional neural network architecture is the vanishing or exploding of the gradient. When increasing the network layers, the Sigmoid and Tanh activation functions have vanished and exploded gradient problems, but they still have practical value in some environments [16]. The ReLU is unbounded above, bounded below, which has constant derivatives for positive inputs that

The associate editor coordinating the review of this manuscript and approving it for publication was Rongbo Zhu.

can avoid the vanishing gradient problem and has faster calculation speed, making the ReLU the most popular activation function [17]; however, the ReLU has the problem of dying neurons, a neuron with all corresponding nodes were dead when it gets zero gradients, and the neurons will not resurrect. For solving the dead neuron problem, the LeakyReLU introduces a parameter with a small value on the negative inputs. That makes the negative gradient not zero and fixes the dying neuron problem. However, which is not differentiable at zero, an improved variational version named DReLU [18] performs better than ReLU and LeakyReLU. The later proposed activation functions like ELU, Swish, and Mish improved the problem of discontinuous differentiability and got better performance for sophisticated convolutional neural networks. The ELU is monotonic and approaches a constant value for negative inputs. Theoretically, the monotonic activation function should get better performance [19], [20], [21]. ZHENG et al. [22] proposed an improved activation function based on ELU and named FELU, which is also monotonic, while the Swish and Mish are not monotonic, and both approach zero when increasing the negative inputs. Some studies tend to use non-monotonic activation functions [23], [24], [25], [26], [27], [28]. There are new proposed activation functions named RMAF [29] and REU (PREU) [30], which are similar to Swish but get better accuracy in their experiments and are also non-monotonic.

This paper proposed a new activation function called PolyLU, which meets the basic principles like continuously differentiable, unbounded for positive inputs, bounded for negative inputs, approximate identity near the origin, smooth, monotonic, and zero-centered. However, some recent newly activation functions like Swish and Mish tend to use the non-monotonic function for negative inputs; the experiment results show that monotonic functions still have their superiority. Those exponential terms are not used in PolyLU. Instead, this paper uses polynomial terms to reduce the computational complexity. The experiments show that the PolyLU has improved some network complexity and has better accuracy over MNIST, Kaggle Cats and Dogs, CIFAR-10, and CIFAR-100 datasets. Test by the CIFAR-100 dataset with batch normalization, PolyLU improves by 0.62%, 2.82%, 2.44%, 1.33%, 2.08%, and 4.26% of accuracy than ELU, Swish, Mish, Leaky ReLU, ReLU, and Tanh respectively. Test by the CIFAR-100 dataset without batch normalization, PolyLU improves by 1.24%, 4.39%, 2.12%, 5.43%, 15.51%, and 8.10% of accuracy than ELU, Swish, Mish, Leaky ReLU, ReLU, and Tanh respectively. In some training models, the Sigmoid activation function cannot converge without the batch normalization layers. The PolyLU does not need the batch normalization layers and still performs better, saving much computing time.

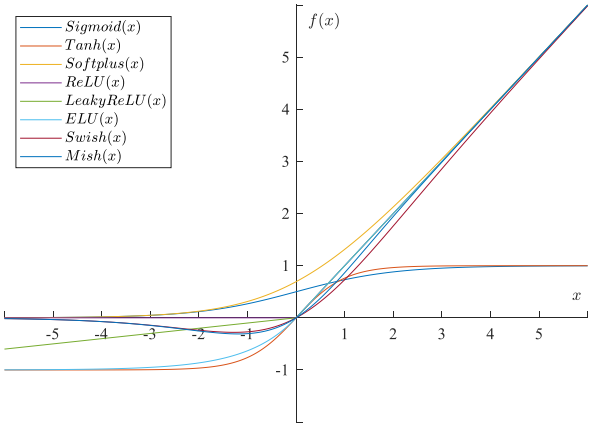

FIGURE 1. The curve of activation functions.

The arrangement of this paper is as follows: Section II introduces the related work, Section III is the proposed activation function, Section IV illustrates the experiments and results, and the conclusion is in the final section, which is Section V.

II. RELATED WORK

The related work section will introduce commonly used activation functions like Sigmoid, Tanh, Softplus, ReLU, Leaky ReLU, ELU, Swish, and Mish. The newly proposed PolyLU activation function will be compared with the aforementioned activation functions except for Softplus. Figure 1 shows the curve of those activation functions, and Figure 2 shows their derivatives.

A. SIGMOID

The Sigmoid is considered the earliest successful activation function; although it is rarely used now, it still has its practical value in binary classification. Eq. 1 shows the Sigmoid activation function and its derivative in Eq. 2.

$$\mathrm{F}_{\mathrm{Sigmoid}}(x) = \frac{1}{1+e^{-x}} \tag{1}$$

$$\mathrm{F'}_{\mathrm{Sigmoid}}(x) = \frac{e^{-x}}{\left(1+e^{-x}\right)^2} \tag{2}$$

The Sigmoid activation function is bounded for both the positive and negative inputs. It tends to quickly saturate when increasing positive and negative inputs, leading to vanishing gradient problems. The exponential terms of Sigmoid increase the complexity of the calculation, and it does not approximate identity near the origin, so one must be careful when choosing the initial weights [70].

B. HYPERBOLIC TANGENT

Eq. 3 shows the Tanh activation function and Eq. 4 is its derivative.

$$\mathrm{F}_{\mathrm{Tanh}}(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3}$$

$$\mathrm{F'}_{\mathrm{Tanh}}(x) = 1 - \frac{\left(e^{x}-e^{-x}\right)^2}{\left(e^{x}+e^{-x}\right)^2} \tag{4}$$

The output range of the Tanh activation function is from a negative to a positive one. Like Sigmoid, Tanh is

bounded for the positive and negative inputs, and it tends to quickly saturate when increasing the positive and negative inputs, leading to the vanishing gradient problem. The Tanh features an approximate identity near the origin and is zero-centered, much better than the Sigmoid. However, the equation structure is more complicated than Sigmoid, making high computational energy consumption.

C. SOFTPLUS

Eq. 5 shows the Softplus activation function, and Eq. 6 is its derivative.

$$\mathrm{F}_{\mathrm{Softplus}}(x) = \ln\left(1 + e^{x}\right) \tag{5}$$

$$\mathrm{F'}_{\mathrm{Softplus}}(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

The Softplus activation function has an output range from zero to infinity, like the combination of ReLU and Sigmoid. The unbounded above property of Softplus is similar to ReLU, while the non-zero output for zero input property is similar to Sigmoid. In the experiment, the output is shifted to zero when the input is zero by subtracting a constant value of $\ln(2)$; since the result is zero when $f(0) = \ln\left(1 + e^{0}\right) - \ln(2) = 0$, then it became zero-centered, the new modified Softplus can be called SoftMinus. The performance of the SoftMinus is just slightly better than Tanh, but worse than the others.

The training accuracy of SoftMinus was higher than ReLU and LeakyReLU, while the evaluation accuracy was not, which somehow explained the superiority of the identity function for positive inputs.

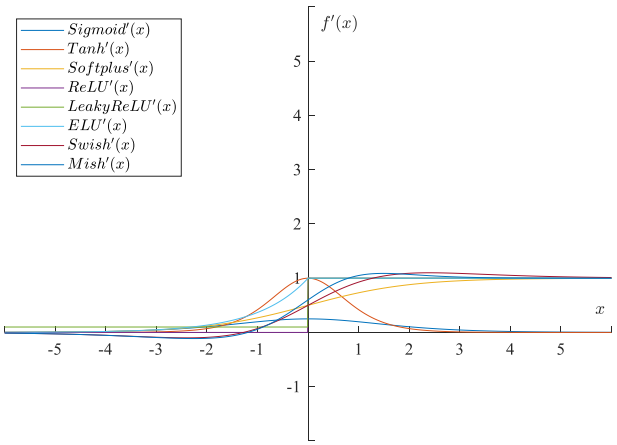

FIGURE 2. The derivative curve of activation functions.

D. ReLU

The ReLU is considered the most successful and popular activation function for its simplicity, efficiency, and fastest calculation speed in both forward and backward propagation since it is an identity function for the positive inputs and its derivative is a constant value of one, the value is always zero for negative inputs. Eq. 7 shows the ReLU activation function, and Eq. 8 is its derivative.

$$\mathrm{F}_{\mathrm{ReLU}}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{7}$$

$$\mathrm{F'}_{\mathrm{ReLU}}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{8}$$

The ReLU activation function is monotonic and unbounded for the positive inputs, which can avoid the vanishing gradient problem. The success of ReLU confirms the superiority of the identity function, and the later proposed successful activation function also follows this principle. The most significant advantage of the ReLU is that it has the lowest computational energy consumption in the existing activation functions. The negative inputs are forced to zero, making the ReLU has a spars feature, which also causes the dying neuron problem. The ReLU is not zero centered, is not differentiable at zero, and is not approximate identity near the origin.

E. LEAKY ReLU

Eq. 9 shows the Leaky ReLU activation function, and Eq. 10 is its derivative.

$$\mathrm{F}_{\mathrm{LeakyReLU}}(x) = \begin{cases} \alpha x & \text{if } x < 0, \alpha \geq 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{9}$$

$$\mathrm{F'}_{\mathrm{LeakyReLU}}(x) = \begin{cases} \alpha & \text{if } x < 0, \alpha \geq 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{10}$$

The Leaky ReLU activation function is an improved version of ReLU. It is zero-centered, strictly monotonic, and unbounded for positive and negative inputs. The non-zero for negative inputs makes LeakyReLU avoid the vanishing gradient problem, and the simple constant parameter multiplied by input only slightly increases the computation time than ReLU. The LeakyReLU performs better than ReLU most of the time.

The Leaky ReLU is continuous but is not differentiable at zero. It needs to choose proper initial weights carefully since there are some differences when the inputs are close to zero.

Compared to the other activation functions, the most significant difference of LeakyReLU is the feature of unbounded below. It is inappropriate that the negative inputs have too much influence on the weights.

F. ELU

Eq. 11 shows the ELU activation function, and Eq. 12 is its derivative. The ELU activation function improved the shortcomings of ReLU and LeakyReLU. The ELU is an approximate identity near the origin. Its derivative and itself are monotonic if the parameter α is set to one. The ELU also has the feature of zero-centered, unbounded for positive inputs, and the minimum value for negative inputs is minus α, which is smooth and differentiable everywhere. Theoretically, the ELU also meets all the basic requirements of an activation function, while the exponential terms of the

ELU and the subtraction calculation take more computational energy consumption.

$$F_{ELU}(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x < 0, \alpha \geq 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{11}$$

$$F'_{ELU}(x) = \begin{cases} \alpha e^x & \text{if } x < 0, \alpha \geq 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{12}$$

Another serious problem with the exponential term is that Euler's number e is an infinite decimal number. It is an approximation but not an exact value when computing, and the calculation result might be slightly different because of the calculation accuracy between other operating systems. When using the Taylor polynomial to estimate the e^x, the error between the actual value of the e^x and the estimation value will increase as the x gets larger and larger. Another problem is that the items of the polynomial are limited by the computing ability of the computer, causing the result to be different between different computers.

G. SWISH

Eq. 13 shows the Swish activation function, and Eq. 14 is derivative. The Google team proposed the Swish activation function in 2017; it can be expressed as the Sigmoid multiplied by the input x. The equation of the Swish is $f(x) = xsigmoid(x)$, that improves all the defects of the Sigmoid. The Swish is smooth, continuously differentiable, unbounded for positive inputs, bounded for negative inputs, and which is approximate identity near the origin., The significant difference between Swish and ELU is the monotonicity; the Swish is not monotonic for negative inputs.

Unlike the ReLU, LeakyReLU, ELU, and Swish do not have an identical function for positive inputs. However, these experiments show that the performances of Swish were not better than ReLU and LeakyReLU when there are not so many network layers.

In deeper neural network architectures and complex datasets, the performance of Swish is better than ReLU, while Swish needs much more computational energy consumption due to the more complex exponential terms.

$$F_{Swish}(x) = \frac{x}{1 + e^{-x}} \tag{13}$$

$$F'_{Swish}(x) = \frac{xe^{-x} + e^{-x} + 1}{(1 + e^{-x})^2} \tag{14}$$

H. MISH

Eq. 15 shows the Mish activation function, and Eq. 16 is its derivative.

$$F_{Mish}(x) = x \tanh\left(\ln\left(1 + e^x\right)\right) \tag{15}$$

$$F'_{Mish}(x) = \frac{e^x \omega}{\delta^2} \tag{16}$$

where the parameter $\omega = 4(x+1) + 4e^{2x} + e^{3x} + e^x(4x+6)$, and parameter $\delta = 2e^x + e^{2x} + 2$.

The Mish activation function is similar to Swish; both are continuously differentiable, unbounded for positive inputs, bounded for negative inputs, and non-monotonic. The Mish is also not an identical function for positive inputs. However, the Mish is closer to the identity function than Swish. The slope of Mish is closer to one and is more significant than Swish for positive inputs.

The Mish is more complicated in calculation than the Swish; the Mish needs more computing resources. This experiment used a smaller batch size because Mish will crash with a larger batch size.

Newly proposed activation functions SAAF [31] and RSigELU [32] perform better than Swish, but they all include more complex exponential terms that increase computational complexity. The tanhLU [33] is similar to Tanh, and the Smish [34] is similar to Mish; both are better than Swish and have complex exponential terms. The non-monotonic activation function EANAF [35], compared with Swish and ReLU using the YOLO v4 [36] model, gets better accuracy and has exponential terms. The PFLU and FPFLU [37] are non-monotonic, without exponential terms, and have a square root in the denominator, increasing the computational complexity. Venkatappareddy et al. [38] proposed a polynomial-based activation function; Zhu et al. [39] proposed a non-monotonic activation function called Logish; both show high accuracy on the CIFAR-10 dataset. TanhSoft-1, TanhSoft-2, and TanhSoft-3 [40] are combinations of Tanh; the exponential terms also increase the computational complexity. Alkhouly et al. [41] proposed the IpLU and AbsLU without exponential terms to make it simple in computation. Thirumagal and Saruladha [42] proposed the EMSLU activation function tested by GAN [43] and performed better than ReLU and ELU, which also has more complex exponential terms. The NAF [44] is used in RNN; the performance is slightly better than ELU, but the operation is complex and requires careful selection of parameters. The PFpM [45] is used in CNN and performs better than Mish and Swish, but its formula is even more complex than Mish and Swish. The AOAF [46] is a slight modification of ReLU; the structure is simple, but the effect is insignificant. Liu et al. [47] proposed the CroReLU for detecting lung cancer, only compared with ReLU. Kalaiselvi et al. [48] proposed the E-Tanh; it only performs better in shallow models. Job et al. [49] proposed the FReLU, which is only suitable for specific hidden layers. Hu et al. [50] alleviate the vanishing gradient for ReLU and Sigmoid by replacing the original derivate with an artificial one. Kiliçarslan et al. [51] proposed the hybrid KAF+RSigELUS and KAF+RSigELUD, which also have complex exponential terms. The ReLTanh [52] combining linear and non-linear terms can mitigate the vanishing gradient problem. Yang et al. [53] proposed the SwishX for enriching activation in lightweight networks. Hossain et al. [54] proposed the LP-ReLU, like a segmented ReLU. The SPOCU [55] does not have exponential terms, but the whole calculation is still

complicated. Chen et al. [56] proposed a triple activation function; it is simple enough, but the performance could be better. Qin et al. [57] proposed the Isigmoid activation function for overcoming the vanishing gradient problem. Tran et al. [58] proposed the Triple-Sigmoid to replace Sigmoid and Softmax. Yuen et al. [59] proposed the UAF, which needs to adjust the parameters and is more time-consuming. The NMAF [60] performs better than ELU, but the structure is more complex than ELU. The OP-Tanish [61] gets high accuracy but a long calculation time. Hwang et al. [62] proposed a universal activation function, which exhibits the properties of activation functions such as ReLU, Sigmoid, and Swish by adjusting three hyperparameters. The RAO [63] is another reconfigurable activation function exhibiting properties like ReLU, Sigmoid, and Tanh. The SupEx [64], like a combination of TanhExp [65] and a reversed ReLU, performs better than other activation functions with four-layer CNN architecture. The HcLSH [66] takes longer computation time for better accuracy and is suitable for deeper architectures and more complex data sets. The NIPUNA [67] is similar to Swish. Vallés-Pérez et al. [68] proposed a straightforward activation function that is helpful for hardware implementations. Unlike most activation functions, its output is positive when the input is negative. Bhimavarapu et al. [69] proposed a less complex activation function, which is cosine-based and not continuous.

III. PROPOSED ACTIVATION FUNCTION (PolyLU)

This section will introduce the new activation function PolyLU. The PolyLU follows all the basic principles that are continuously differentiable, smooth, unbounded for positive inputs, bounded for negative inputs, approximate identity near the origin, zero-centered, and monotonic. Table 1 compares the specific features of PolyLU with those activation functions mentioned in section II.

TABLE 1. Comparison of specific features.

Function	smooth	unbounded positive	zero centered	monotonic
Sigmoid	Yes	No	No	Yes
Tanh	Yes	No	Yes	Yes
Softplus	Yes	Yes	No	Yes
ReLU	No	Yes	No	Yes
LeakyReLU	No	Yes	Yes	Yes
ELU	Yes	Yes	Yes	Yes
Swish	Yes	Yes	Yes	No
Mish	Yes	Yes	Yes	No
PolyLU	Yes	Yes	Yes	Yes

Most of the activation functions use exponential terms of Euler's number e, e is an irrational constant, and the definition of e is shown in Eq. 17; the working environment will limit the practical value of n or m, the actual value of e used in calculation depends on the precision of the computer system, and that might be different for different computers. The Maclaurin series of e^x is shown in Eq. 18; again, the actual maximum value of n is also constrained by the computing system.

$$e = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n \text{ or } e = \lim_{m\to 0}(1+m)^{\frac{1}{m}} \tag{17}$$

$$e^x = \sum_{n=0}^{\infty}\frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} \tag{18}$$

The derivative of e^x is itself; that is the advantage of using e^x, while in the practical calculation, the e^x can only be estimated, and the definite value cannot be obtained. When x increases, the error between the actual and the estimation values increases and is limited by the computing ability. Different computer calculation results may be different, and it takes more time to calculate exponential terms.

Considering those disadvantages when using terms of e^x, this paper proposed a polynomial term to replace the exponential term with a similar curve shape as ELU but has a relatively small slope for negative inputs.

Eq. 19 shows the PolyLU activation function, which is precisely the same as ReLU and ELU for positive inputs, and that keeps all the superiority of the identity function.

$$\mathrm{F}_{\mathrm{PolyLU}}(x) = \begin{cases} \dfrac{1}{1-x} - 1 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{19}$$

Although the curves of PolyLU and ELU are similar and very close, these experiments show that this slight difference significantly impacts performance. The PolyLU got 1.24% (without batch normalization), 0.62% (with batch normalization), and 0.07% performance better than ELU in validation accuracy with those datasets CIFAR-100, CIFAR-10, and Cats and Dogs, respectively.

The derivative of PolyLU for positive inputs is a constant one. In the negative parts, the derivative of the PolyLU can be obtained by the following:

$$\frac{d}{dx}\left(\frac{1}{1-x} - 1\right) = \frac{1}{(1-x)^2} \tag{20}$$

Finally, the derivative of the PolyLU is Eq. 21, and Figure 3 is the curve of the PolyLU and its derivative.

$$\mathrm{F'}_{\mathrm{PolyLU}}(x) = \begin{cases} \dfrac{1}{(1-x)^2} & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{21}$$

For positive inputs, the PolyLU is the same as ReLU and ELU, they all use an identity map, and the feature of identity reduces the probability of gradient disappearance and performs better.

For negative inputs, instead of exponential terms, this paper used a polynomial term, which is the reciprocal of $1-x$, and that needs less computational energy consumption.

When inputs turn negative, use the reciprocal of (1-x) to replace the ex of ELU; the (1-x) is a definite value rather than the approximated ex, and the rate of decline of PolyLU is slower than ELU for negative inputs, that means the resolution contributed by the negative value of PolyLU is higher

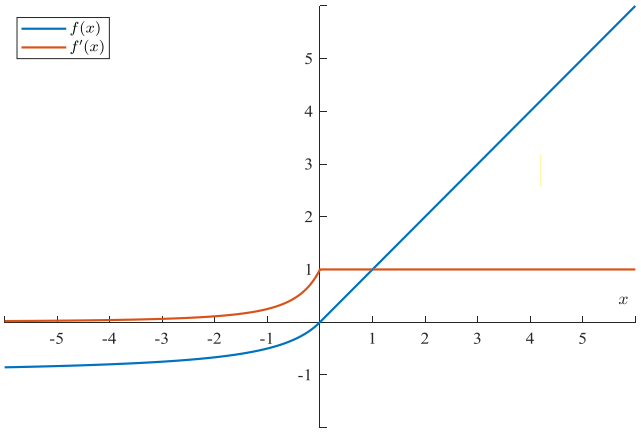

FIGURE 3. The PolyLU f(x) and its derivative f'(x).

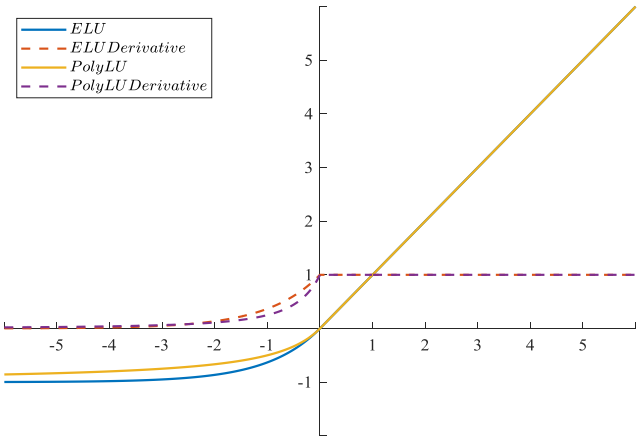

FIGURE 4. The difference between ELU and PoluLU.

than ELU, Figure 4 indicates the difference between ELU and PolyLU.

IV. EXPERIMENTS AND RESULTS

The experiments were carried out with two different personal computers. The first has only one GeForce GTX1050Ti GPU, with Intel i7-8700 CPU @3.20GHz and 16.0 GB RAM. The second one has two GeForce RTX2080Ti GPUs, Intel i7-9700 CPU @3.00GHz, and 64.0 GB RAM. The operating system is Windows 10 for both computers.

The frameworks used in this paper are TensorFlow 1.13 and TensorFlow 2.2 with Keras API. The TensorFlow 1.13 with the MNIST dataset was used to implement the custom mathematical functions for comparing the time consumption of PolyLU, Sigmoid, Tanh, ReLU, LeakyReLU, ELU, Swish, and Mish activation functions. The TensorFlow 2.2 was used on the CIFAR-10 and the CIFAR-100 datasets to evaluate the performance of PolyLU, Sigmoid, Tanh, ReLU, LeakyReLU, ELU, Swish, and Mish. The Kaggle Cats and Dogs dataset evaluated the performance between PolyLU, modified Softplus, Tanh, ReLU, LeakyReLU, ELU, Swish, and Mish.

The first experiment uses the second i7-9700 computer with the CIFAR-100 dataset; there are sixty thousand 32 by 32 color images with one hundred categories in the CIFAR-100 dataset, five-sixths of them used for training and one-sixth for validation; each class has only 500 samples for training. The computer has two GeForce RTX2080Ti GPUs. Referring to the VGG [71] model, there are eight convolution layers and five fully connected layers. A batch normalization layer follows each convolution layer. Every two convolution layers are followed by an average pooling layer and a dropout layer; the dropout rates are 0.2, 0.3, 0.4, and 0.5, respectively. A batch normalization layer follows the first fully connected layer, and a dropout layer follows the second last fully connected layer; the batch size is 128. The optimizer is Adam [72], the loss function is sparse categorical cross entropy, 20% of the training data were used for validation, and the total parameters are 7,500,324. The trainable parameters are 7,492,900, and the non-trainable parameters are 7,424. Each activation function runs for 500 epochs and five times, taking the average training, maximum, minimum, and average validation accuracy. Figure 5 shows the layers and shapes of the network.

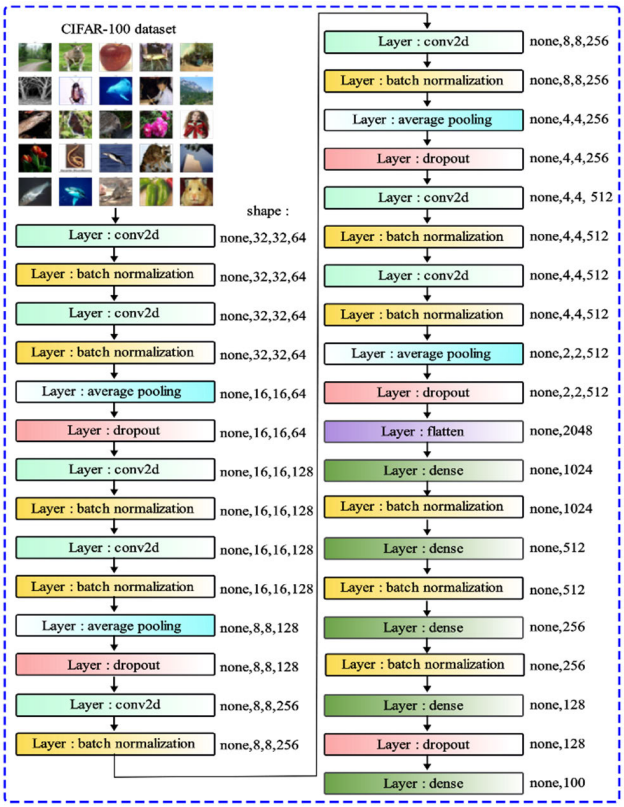

FIGURE 5. Layers and shape of experiment 1.

Then, run the same model for every activation function but remove all batch normalization layers. The same for each activation function runs for 500 epochs and five times,

taking the average training, maximum, minimum, and average validation accuracy.

Table 2 shows the training accuracy, maximum validation accuracy, minimum validation accuracy, and average validation accuracy of each activation function with the CIFAR-100 dataset, with batch normalization.

TABLE 2. Result of CIFAR-100 dataset (with batch normalization).

Function	Average Training accuracy (%)	Maximum validation accuracy (%)	Minimum validation accuracy (%)	Average validation Accuracy (%)
PolyLU	85.53	**65.57**	**65.09**	**65.36**
ELU	85.68	65.41	64.13	64.74
Swish	89.18	62.95	61.54	62.54
Mish	87.20	63.32	62.63	62.91
LeakyReLU	83.64	64.59	63.51	64.03
ReLU	83.89	64.52	62.65	63.28
Tanh	80.02	62.48	60.16	61.10
Sigmoid	8.149	64.65	62.45	63.33

Figure 6 shows the CIFAR-100 training and validation accuracy historical charts with batch normalization for the above activation functions.

Table 3 shows each activation function's training, maximum, minimum, and average validation accuracy with the CIFAR-100 dataset without batch normalization. The Sigmoid cannot run under this kind of model. Without batch normalization, the validation accuracy of PolyLU, ELU, and Mish increased by 1.08%, 0.46%, and 1.41%, respectively, while the others decreased.

TABLE 3. Result of CIFAR-100 dataset (without batch normalization).

Function	Average Training accuracy (%)	Maximum validation accuracy (%)	Minimum validation accuracy (%)	Average validation accuracy (%)
PolyLU	73.91	**67.00**	**66.04**	**66.44**
ELU	67.59	65.61	64.76	65.20
Swish	79.68	63.68	60.18	62.05
Mish	81.37	64.94	63.64	64.32
LeakyReLU	69.07	62.17	60.00	61.01
ReLU	50.44	52.91	48.79	51.28
Tanh	57.53	59.55	57.49	58.34
Sigmoid	-	-	-	-

Figure 7 shows the CIFAR-100 training and validation accuracy historical charts without batch normalization for the above activation functions.

The second experiment uses the CIFAR-10 dataset, which has identical amounts of images as CIFAR-100 but with only ten categories. This experiment used the first i7-8700 computer, eight convolutions, and four fully connected layers. Each convolution layer is followed by one batch normalization layer, a maximum pooling layer, and a dropout layer following every two convolution layers. The dropout rates are 0.2, 0.3, 0.4, and 0.5, respectively. Each fully connected layer is followed by a dropout layer with a rate of 0.5, and the batch size is 256. The optimizer and loss function are

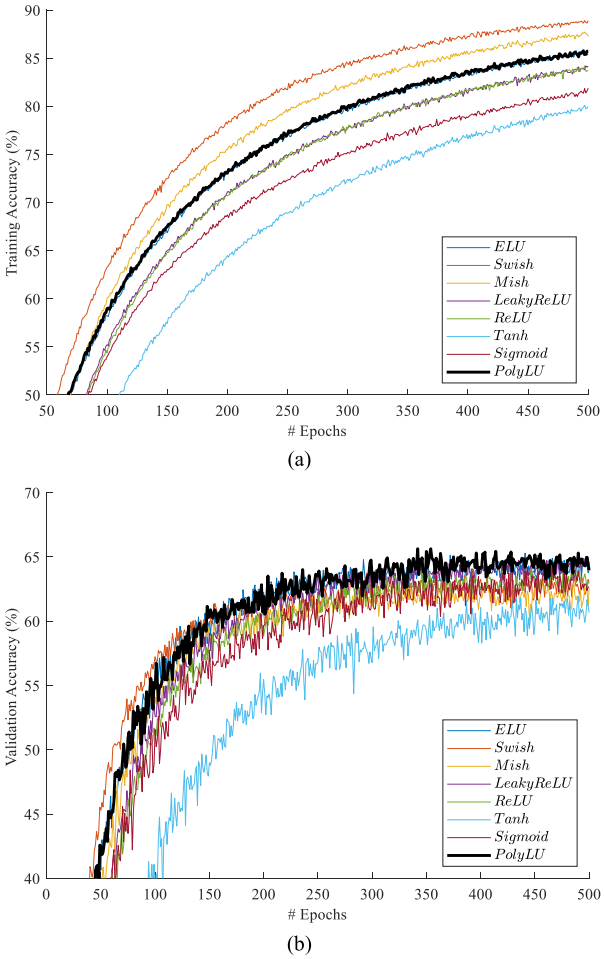

FIGURE 6. CIFAR-100 accuracy with batch normalization, (a) Training, (b) Validation.

identical to the first experiment. The total parameters are 5,907,658, the trainable parameters are 5,903,818, and the non-trainable parameters are 3,840. Each activation function also runs for 300 epochs. Table 4 shows the results of each activation function with the CIFAR-10 dataset, with batch normalization.

TABLE 4. Result of CIFAR-10 dataset (with batch normalization).

Function	Average Training accuracy (%)	Maximum validation accuracy (%)	Minimum validation accuracy (%)	Average validation accuracy (%)
PolyLU	94.50	**91.59**	89.78	**90.46**
ELU	94.89	90.74	**89.98**	90.30
Swish	95.24	90.32	89.30	89.78
Mish	95.26	90.05	88.67	89.51
LeakyReLU	91.24	89.97	88.97	89.43
ReLU	91.64	89.89	88.38	89.17
Tanh	94.11	89.82	88.43	89.20
Sigmoid	90.52	90.25	88.44	89.38

Figure 8 shows the layers and shapes of the network.

This experiment runs the same model for every activation function but takes out all the batch normalization layers; the same for each activation function runs for 500 epochs and five

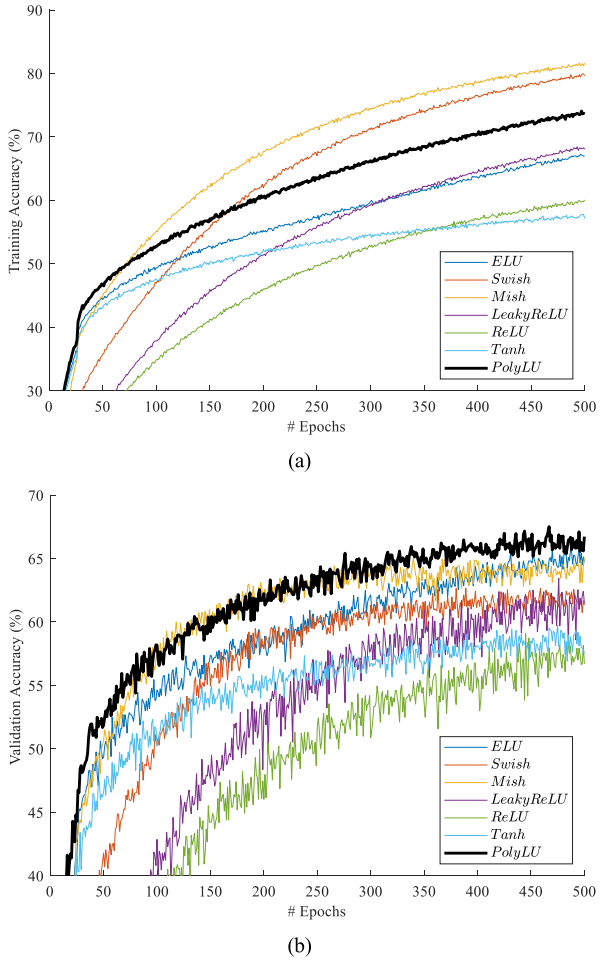

FIGURE 7. CIFAR-100 accuracy without batch normalization, (a) Training, (b) Validation.

TABLE 5. Result of CIFAR-10 dataset (without batch normalization).

Function	Average Training accuracy (%)	Maximum validation accuracy (%)	Minimum validation accuracy (%)	Average validation accuracy (%)
PolyLU	92.00	89.95	**89.06**	**89.61**
ELU	92.35	**90.51**	88.79	89.54
Swish	91.17	87.83	86.78	87.28
Mish	91.11	88.28	87.22	87.69
LeakyReLU	88.41	87.69	85.53	86.24
ReLU	82.93	83.35	79.91	81.75
Tanh	89.66	88.37	84.58	86.97
Sigmoid	-	-	-	-

times, taking the average training accuracy, maximum validation accuracy, minimum validation accuracy, and average validation accuracy. Table 5 shows each activation function's training, maximum, minimum, and average validation accuracy with the CIFAR-10 dataset without batch normalization. Without batch normalization, the validation accuracy of PolyLU and ELU decreased by 0.85% and 0.77%, while the others decreased by more than two times. Again, the Sigmoid cannot converge under this model.

TABLE 6. The performance with cats and dogs dataset.

Function	Average Training accuracy (%)	Maximum validation accuracy (%)	Minimum validation accuracy (%)	Average validation accuracy (%)
PolyLU	99.56	**97.49**	95.94	**96.87**
ELU	99.61	97.35	95.71	96.80
Swish	99.54	97.23	96.00	96.68
Mish	99.61	96.97	96.15	96.60
LeakyReLU	99.56	96.96	95.99	96.54
ReLU	99.57	97.15	**96.30**	96.71
Tanh	99.43	96.78	94.97	96.17
Sigmoid	99.14	96.38	76.09	89.84

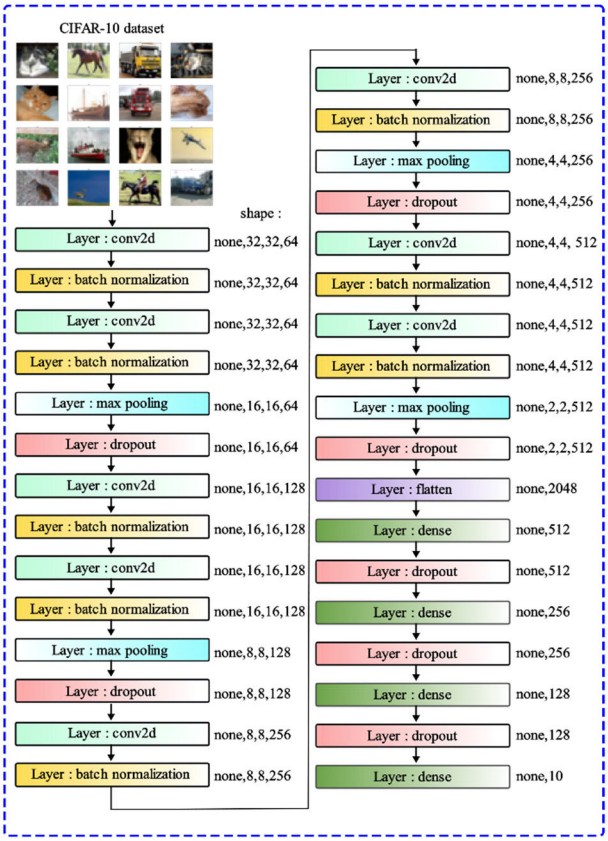

FIGURE 8. Layers and shapes of experiment 2.

Figure 9 is the CIFAR-10 training and validation accuracy historical charts with batch normalization for the above activation functions.

Figure 10 shows the CIFAR-10 training and validation accuracy, training, and validation loss history charts without batch normalization for the above activation functions. This experiment did not include the Sigmoid activation function because it cannot converge without batch normalization layers. The ReLU and The LeakyReLU show violent shock under specific training models, while PolyLU is more stable than the above. The training loss of the ReLU was getting higher when the epochs were over 150 because of overfitting.

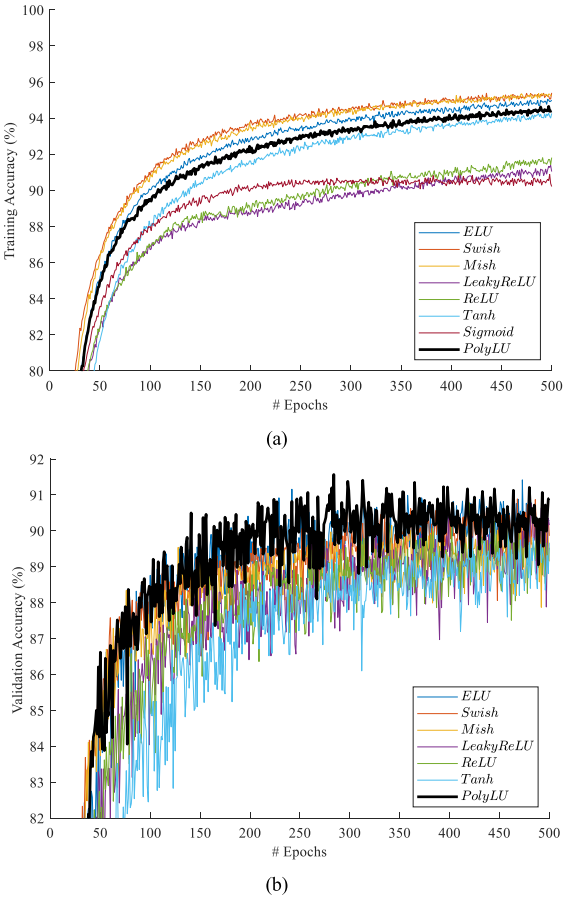

FIGURE 9. CIFAR-10 accuracy with batch normalization, (a) Training, (b)Validation.

The third experiment used the same hardware as the second experiment and used the Kaggle Cats and Dogs dataset. There are 11 convolution layers; every two are followed by a maximum pooling layer except the first two and the last convolution layers. There are no dropout and no batch normalization layers. Referring to the residual network [73], implement a back residual to the network architecture. This experiment found that adding the batch normalization layers does improve the performance of the Swish, Mish, Leaky ReLU, ReLU, and all the other activation functions in this experiment. The batch normalization layers have no significant influence on the PolyLU. Not using batch normalization can significantly reduce computational complexity. Table 6 shows the results of each activation function with the Cats and Dogs dataset.

Table 7 compares the validation accuracy standard deviation for all the activation functions in the above experiments. The standard deviation σ is calculated according to Eq. 22. Table 8 shows the result of the One-way ANOVA for the above experiments.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{5}(x_i - \bar{x})^2}{5}} \tag{22}$$

TABLE 7. The standard deviation of validation accuracy.

Function	CIFAR 100 with batch	CIFAR 100 without batch	CIFAR 10 with batch	CIFAR 10 without batch	Cats and Dogs
PolyLU	0.0022	0.0035	0.0061	0.0030	0.0052
ELU	0.0049	0.0031	0.0029	0.0069	0.0058
Swish	0.0051	0.0115	0.0032	0.0038	0.0056
Mish	0.0027	0.0049	0.0054	0.0041	0.0031
LeakyReLU	0.0045	0.0080	0.0021	0.0078	0.0035
ReLU	0.0065	0.0155	0.0050	0.0120	0.0028
Tanh	0.0076	0.0084	0.0045	0.0138	0.0066
Sigmoid	0.0082	-	0.0072	-	0.0763

TABLE 8. The One-way ANOVA of the above experiments; SS is the sum of squares for each source of variation; df is the degree of freedom; MS is the mean sum of squares; F is the overall F-value, calculated as MS between / MS within; The P-value corresponding to the overall F-value; The F cri. value that corresponds to $\alpha = 0.5$.

Source	CIFAR 100 with batch	CIFAR 100 without batch	CIFAR 10 with batch	CIFAR 10 without batch	Cats and Dogs
SS	0.0061	0.0802	0.0008	0.0210	0.02028
df	7	6	7	6	7
MS	0.00088	0.01337	0.00012	0.0035	0.002898
F	22.568	135.207	3.8628	40.548	3.09826
P-value	1.1×10^{-10}	2.3×10^{-19}	0.00374	1.5×10^{-12}	0.01306
F cri.	2.3127	2.4453	2.3127	2.4453	2.3127

TABLE 9. The autoencoder with the ECG5000 dataset.

Function	Average accuracy (%)	Average precision (%)	Average recall (%)
PolyLU	**95.98**	99.4297	**93.3571**
ELU	95.80	99.3897	93.0714
Swish	95.26	**99.6128**	91.8929
Mish	95.52	99.5766	92.3929
LeakyReLU	95.16	99.6121	91.7143
ReLU	94.76	99.6091	91.0000
Tanh	95.46	99.2725	92.5714
Sigmoid	94.50	99.2203	90.8929

where x_i is the validation accuracy of each time, and $\bar{x}$ is the average of five times. Each activation function runs five times.

The fourth experiment is based on the Tensorflow tutorials; it trains an autoencoder to detect anomalies on the ECG5000 dataset. This dataset contains 5000 electrocardiograms; each electrocardiogram has 140 data points. The whole process is carried out five times for each activation function. Table 9 shows the results of accuracy, precision, and recall.

In the fifth experiment, the aim is to compare the time consumption between those activation functions rather than improve validation or evaluation accuracy, so it just designed a simple convolution layer network, each followed by a maximum pooling layer, using one dropout at the end, and tested with the MNIST dataset. Since the PolyLU is not a built-in function of TensorFlow, it needs to define the custom Python function to replace the built-in activation functions

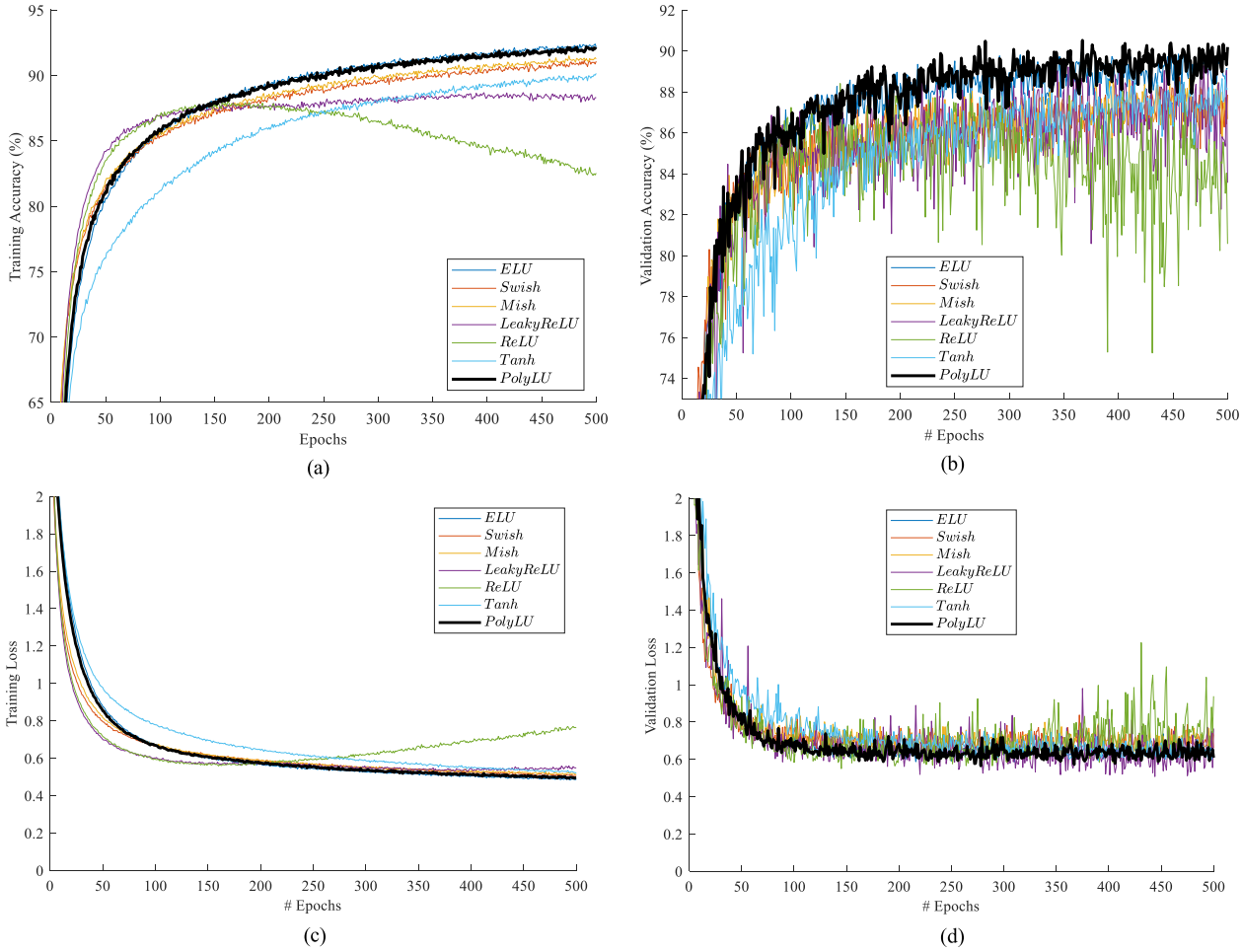

FIGURE 10. CIFAR-10 without batch normalization, (a) Training accuracy, (b) Validation accuracy, (c)Training loss, (d)Validation loss.

TABLE 10. Time consumption with the MNIST dataset.

Function	Evaluation accuracy (%)	Step/sec	Time/epoch
PolyLU	98.19	2.2070	**45.307**
ELU	98.30	1.0590	94.433
Swish	98.18	0.2549	392.312
LeakyReLU	98.13	3.0151	33.166
ReLU	98.01	3.0547	32.737
Tanh	98.08	0.3941	253.725
Sigmoid	89.27	0.3304	302.628

of TensorFlow and compare the time consumption under the same conditions. This experiment defines custom functions for the above activation functions except for Mish because it is too complicated and will take too long to calculate for custom backward propagation; it is the most time-consuming activation function of the above. In this experiment, Mish is not included in the comparison. Table 10 illustrates the evaluation accuracy, step per second, and time consumption per epoch of each activation function with the MNIST dataset using the custom-defined forward and backward propagation function.

As shown in Table 10, the operation time of PolyLU is slower than ReLU and LeakyReLU but is much faster than the others; the Swish is 8.65 times slower than PolyLU; this phenomenon shows that without using the exponential terms can speed up the calculation of the convolutional neural network.

V. CONCLUSION

This paper proposed a new activation function named PolyLU, which satisfies all the following basic principles for the activation function: approximate identity near the origin, unbounded for positive inputs, bounded for negative inputs, continuously differentiable, smooth, monotonic and zero centered, and there is no exponential term in PolyLU, that reduce the computational complexity. Compared to those common activation functions like Sigmoid, Tanh, ReLU, LeakyReLU, ELU, Mish, and Swish, the experiments show that the PolyLU has improved some network complexity and has better accuracy over MNIST, Kaggle Cats and Dogs, CIFAR-10 and CIFAR-100 datasets. Test by the CIFAR-100 dataset with batch normalization, PolyLU improves by 0.62%, 2.82%, 2.44%, 1.33%, 2.08%, and 4.26% of accuracy than ELU,

Swish, Mish, Leaky ReLU, ReLU, and Tanh respectively. Test by the CIFAR-100 dataset without batch normalization, PolyLU improves by 1.24%, 4.39%, 2.12%, 5.43%, 15.51%, and 8.10% of accuracy than ELU, Swish, Mish, Leaky ReLU, ReLU, and Tanh respectively. The operation speed of the PolyLU is slower than ReLU and LeakyReLU, while it is much faster than the others. All the above characteristics illustrate the superiority of PolyLU; in some situations, the PolyLU provides another better option in the convolutional neural network. This research focuses mainly on the CNN network; future work is to implement this activation function in other networks.

REFERENCES

[1] B. C. Csáji, "Approximation with artificial neural networks," M.S. thesis, Dept. Sci., Eötvös Loránd Univ., Hungary, 2001, vol. 24, no. 48, p. 7.

[2] T. Chen and H. Chen, "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems," *IEEE Trans. Neural Netw.*, vol. 6, no. 4, pp. 911–917, Jul. 1995.

[3] Y. Lu and J. Lu, "A universal approximation theorem of deep neural networks for expressing probability distributions," 2020, *arXiv:2004.08867*.

[4] D. A. Winkler and T. C. Le, "Performance of deep and shallow neural networks, the universal approximation theorem, activity cliffs, and QSAR," *Mol. Informat.*, vol. 36, nos. 1–2, Jan. 2017, Art. no. 1600118.

[5] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals, Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[6] T. Chen, H. Chen, and R.-w. Liu, "A constructive proof and an extension of Cybenko's approximation theorem," in *Computing Science and Statistics*. Cham, Switzerland: Springer, 1992, pp. 163–168.

[7] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[8] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[9] A. N. Gorban and D. C. Wunsch, "The general approximation theorem," in *Proc. IEEE Int. Joint Conf. Neural Netw., IEEE World Congr. Comput. Intell.*, May 1998, pp. 1271–1274.

[10] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, Jun. 2011, pp. 315–323.

[11] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 1–8.

[12] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, 2013, pp. 1–6.

[13] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*.

[14] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*.

[15] D. Misra, "Mish: A self regularized non-monotonic activation function," 2019, *arXiv:1908.08681*.

[16] M. A. Mansor and S. Sathasivam, "Performance analysis of activation function in higher order logic programming," *AIP Conf. Proc.*, vol. 1750, no. 1, Jun. 2016, Art. no. 030007.

[17] R. Parhi and R. D. Nowak, "The role of neural network activation functions," *IEEE Signal Process. Lett.*, vol. 27, pp. 1779–1783, 2020.

[18] X. Hu, P. Niu, J. Wang, and X. Zhang, "A dynamic rectified linear activation units," *IEEE Access*, vol. 7, pp. 180409–180416, 2019.

[19] R. J. Williams, "The logic of activation functions," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA, USA: MIT Press, 1986, pp. 423–443.

[20] S. Ravanbakhsh, B. Póczos, J. Schneider, D. Schuurmans, and R. Greiner, "Stochastic neural networks with monotonic activation functions," in *Proc. 19th Int. Conf. Artif. Intell. Statist.*, 2016, pp. 809–818.

[21] I. S. Isa, Z. Saad, S. Omar, M. K. Osman, K. A. Ahmad, and H. A. M. Sakim, "Suitable MLP network activation functions for breast cancer and thyroid disease detection," in *Proc. 2nd Int. Conf. Comput. Intell., Modeling Simulation*, Sep. 2010, pp. 39–44.

[22] Z. Qiumei, T. Dan, and W. Fenghua, "Improved convolutional neural network based on fast exponentially linear unit activation function," *IEEE Access*, vol. 7, pp. 151359–151367, 2019.

[23] J. M. Sopena, E. Romero, and R. Alquezar, "Neural networks with periodic and monotonic activation functions: A comparative study in classification problems," in *Proc. 9th Int. Conf. Artif. Neural Netw.*, Sep. 1999, pp. 323–328.

[24] 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," *Neural Netw.*, vol. 79, pp. 117–127, Jul. 2016.

[25] K. W. Wong, C. S. Leung, and S. J. Chang, "Use of periodic and monotonic activation functions in multilayer feedforward neural networks trained by extended Kalman filter algorithm," *IEE Proc. Vis., Image Signal Process.*, vol. 149, no. 4, pp. 217–224, Aug. 2002.

[26] E. Liz and A. Ruiz-Herrera, "Global dynamics of discrete neural networks allowing non-monotonic activation functions," *Nonlinearity*, vol. 27, no. 2, pp. 289–304, Feb. 2014.

[27] F. A. Khanday, N. A. Kant, and M. R. Dar, "Low-voltage realization of neural networks using non-monotonic activation function for digital applications," *Recent Adv. Electr. Electron. Eng., Formerly Recent Patents Electr. Electron. Eng.*, vol. 11, no. 3, pp. 367–375, Jul. 2018.

[28] N. Patwardhan, M. Ingalhalikar, and R. Walambe, "ARiA: Utilizing Richard's curve for controlling the non-monotonicity of the activation function in deep neural nets," 2018, *arXiv:1805.08878*.

[29] Y. Yu, K. Adu, N. Tashi, P. Anokye, X. Wang, and M. A. Ayidzoe, "RMAF: Relu-Memristor-like activation function for deep learning," *IEEE Access*, vol. 8, pp. 72727–72741, 2020.

[30] Y. Ying, J. Su, P. Shan, L. Miao, X. Wang, and S. Peng, "Rectified exponential units for convolutional neural networks," *IEEE Access*, vol. 7, pp. 101633–101640, 2019.

[31] Y. Zhou, D. Li, S. Huo, and S.-Y. Kung, "Shape autotuning activation function," *Expert Syst. Appl.*, vol. 171, Jun. 2021, Art. no. 114534, doi: 10.1016/j.eswa.2020.114534.

[32] S. Kiliçarslan and M. Celik, "RSigELU: A nonlinear activation function for deep neural networks," *Expert Syst. Appl.*, vol. 174, Jul. 2021, Art. no. 114805, doi: 10.1016/j.eswa.2021.114805.

[33] S.-L. Shen, N. Zhang, A. Zhou, and Z.-Y. Yin, "Enhancement of neural networks with an alternative activation function tanhLU," *Expert Syst. Appl.*, vol. 199, Aug. 2022, Art. no. 117181, doi: 10.1016/j.eswa.2022.117181.

[34] X. Wang, H. Ren, and A. Wang, "Smish: A novel activation function for deep learning methods," *Electronics*, vol. 11, no. 4, p. 540, Feb. 2022, doi: 10.3390/electronics11040540.

[35] E. Chai, W. Yu, T. Cui, J. Ren, and S. Ding, "An efficient asymmetric nonlinear activation function for deep neural networks," *Symmetry*, vol. 14, no. 5, p. 1027, May 2022, doi: 10.3390/sym14051027.

[36] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[37] M. Zhu, W. Min, Q. Wang, S. Zou, and X. Chen, "PFLU and FPFLU: Two novel non-monotonic activation functions in convolutional neural networks," *Neurocomputing*, vol. 429, pp. 110–117, Mar. 2021, doi: 10.1016/j.neucom.2020.11.068.

[38] P. Venkatappareddy, J. Culli, S. Srivastava, and B. Lall, "A Legendre polynomial based activation function: An aid for modeling of max pooling," *Digit. Signal Process.*, vol. 115, Aug. 2021, Art. no. 103093, doi: 10.1016/j.dsp.2021.103093.

[39] H. Zhu, H. Zeng, J. Liu, and X. Zhang, "Logish: A new nonlinear nonmonotonic activation function for convolutional neural network," *Neurocomputing*, vol. 458, pp. 490–499, Oct. 2021, doi: 10.1016/j.neucom.2021.06.067.

[40] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey, "TanhSoft—Dynamic trainable activation functions for faster learning and better performance," *IEEE Access*, vol. 9, pp. 120613–120623, 2021, doi: 10.1109/access.2021.3105355.

[41] A. A. Alkhouly, A. Mohammed, and H. A. Hefny, "Improving the performance of deep neural networks using two proposed activation functions," *IEEE Access*, vol. 9, pp. 82249–82271, 2021, doi: 10.1109/access.2021.3085855.

[42] E. Thirumagal and K. Saruladha, "Lung cancer classification using exponential mean saturation linear unit activation function in various generative adversarial network models," *Int. J. Imag. Syst. Technol.*, vol. 32, no. 4, pp. 1414–1428, Jul. 2022, doi: 10.1002/ima.22719.

[43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[44] Q. Zhu and M. Tan, "A novel activation function based recurrent neural networks and their applications on sentiment classification and dynamic problems solving," *Frontiers Neurorobot.*, vol. 16, Sep. 2022, Art. no. 1022887, doi: 10.3389/fnbot.2022.1022887.

[45] A. Mondal and V. K. Shrivastava, "A novel Parametric Flatten-p Mish activation function based deep CNN model for brain tumor classification," *Comput. Biol. Med.*, vol. 150, Nov. 2022, Art. no. 106183, doi: 10.1016/j.compbiomed.2022.106183.

[46] Y. Jiang, J. Xie, and D. Zhang, "An adaptive offset activation function for CNN image classification tasks," *Electronics*, vol. 11, no. 22, p. 3799, Nov. 2022, doi: 10.3390/electronics11223799.

[47] Y. Liu, H. Wang, K. Song, M. Sun, Y. Shao, S. Xue, L. Li, Y. Li, H. Cai, Y. Jiao, N. Sun, M. Liu, and T. Zhang, "CroReLU: Cross-crossing space-based visual activation function for lung cancer pathology image recognition," *Cancers*, vol. 14, no. 21, p. 5181, Oct. 2022, doi: 10.3390/cancers14215181.

[48] T. Kalaiselvi, S. T. Padmapriya, K. Somasundaram, and S. Praveenkumar, "E-Tanh: A novel activation function for image processing neural network models," *Neural Comput. Appl.*, vol. 34, no. 19, pp. 16563–16575, Oct. 2022, doi: 10.1007/s00521-022-07245-x.

[49] M. S. Job, P. H. Bhateja, M. Gupta, K. Bingi, and B. R. Prusty, "Fractional rectified linear unit activation function and its variants," *Math. Problems Eng.*, vol. 2022, pp. 1–15, Jun. 2022, doi: 10.1155/2022/1860779.

[50] Z. Hu, J. Zhang, and Y. Ge, "Handling vanishing gradient problem using artificial derivative," *IEEE Access*, vol. 9, pp. 22371–22377, 2021, doi: 10.1109/access.2021.3054915.

[51] S. Kiliçarslan and M. Celik, "KAF+RSigELU: A nonlinear and kernel-based activation function for deep neural networks," *Neural Comput. Appl.*, vol. 34, no. 16, pp. 13909–13923, Aug. 2022, doi: 10.1007/s00521-022-07211-7.

[52] X. Wang, Y. Qin, Y. Wang, S. Xiang, and H. Chen, "ReLTanh: An activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis," *Neurocomputing*, vol. 363, pp. 88–98, Oct. 2019, doi: 10.1016/j.neucom.2019.07.017.

[53] L. Yang, Q. Song, Z. Fan, C. Liu, and M. Hu, "Rethinking the activation function in lightweight network," *Multimedia Tools Appl.*, vol. 82, no. 1, pp. 1355–1371, Jan. 2023, doi: 10.1007/s11042-022-13217-z.

[54] M. T. Hossain, S. W. Teng, F. Sohel, and G. Lu, "Robust image classification using a low-pass activation function and DCT augmentation," *IEEE Access*, vol. 9, pp. 86460–86474, 2021, doi: 10.1109/access.2021.3089598.

[55] J. Kiselák, Y. Lu, J. Švihra, P. Szépe, and M. Stehlík, "'SPOCU': Scaled polynomial constant unit activation function," *Neural Comput. Appl.*, vol. 33, no. 8, pp. 3385–3401, 2020, doi: 10.1007/s00521-020-05182-1.

[56] G. Chen, Q. Wang, X. Li, and Y. Zhang, "Target detection based on a new triple activation function," *Syst. Sci. Control Eng.*, vol. 10, no. 1, pp. 629–635, Dec. 2022, doi: 10.1080/21642583.2022.2091060.

[57] 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," *IEEE Trans. Ind. Electron.*, vol. 66, no. 5, pp. 3814–3824, May 2019, doi: 10.1109/tie.2018.2856205.

[58] D. T. Tran, N. Shimada, and J.-H. Lee, "Triple-sigmoid activation function for deep open-set recognition," *IEEE Access*, vol. 10, pp. 77668–77678, 2022, doi: 10.1109/access.2022.3192621.

[59] B. Yuen, M. T. Hoang, X. Dong, and T. Lu, "Universal activation function for machine learning," *Sci. Rep.*, vol. 11, no. 1, Sep. 2021, doi: 10.1038/s41598-021-96723-8.

[60] A. A. Yahya, K. Liu, A. Hawbani, Y. Wang, and A. N. Hadi, "A novel image classification method based on residual network, inception, and proposed activation function," *Sensors*, vol. 23, no. 6, p. 2976, Mar. 2023, doi: 10.3390/s23062976.

[61] S. Ankalaki and M. N. Thippeswamy, "A novel optimized parametric hyperbolic tangent swish activation function for 1D-CNN: Application of sensor-based human activity recognition and anomaly detection," *Multimedia Tools Appl.*, May 2023, doi: 10.1007/s11042-023-15766-3.

[62] S.-Y. Hwang and J.-J. Kim, "A universal activation function for deep learning," *Comput., Mater. Continua*, vol. 75, no. 2, pp. 3553–3569, 2023, doi: 10.32604/cmc.2023.037028.

[63] A. E. Dehghanpour and S. Koohi, "All-optical recurrent neural network with reconfigurable activation function," *IEEE J. Sel. Topics Quantum Electron.*, vol. 29, no. 2, pp. 1–14, Mar. 2023, doi: 10.1109/jstqe.2022.3173927.

[64] S. Kiliçarslan, C. Közkurt, S. Baş, and A. Elen, "Detection and classification of pneumonia using novel Superior Exponential (SupEx) activation function in convolutional neural networks," *Expert Syst. Appl.*, vol. 217, May 2023, Art. no. 119503, doi: 10.1016/j.eswa.2023.119503.

[65] X. Liu and X. Di, "TanhExp: A smooth activation function with high convergence speed for lightweight neural networks," *IET Comput. Vis.*, vol. 15, no. 2, pp. 136–150, Mar. 2021, doi: 10.1049/cvi2.12020.

[66] H. Abdel-Nabi, G. Al-Naymat, M. Ali, and A. Awajan, "HcLSH: A novel non-linear monotonic activation function for deep learning methods," *IEEE Access*, vol. 11, pp. 47794–47815, 2023, doi: 10.1109/access.2023.3276298.

[67] G. Madhu, S. Kautish, K. A. Alnowibet, H. M. Zawbaa, and A. W. Mohamed, "NIPUNA: A novel optimizer activation function for deep neural networks," *Axioms*, vol. 12, no. 3, p. 246, Feb. 2023, doi: 10.3390/axioms12030246.

[68] 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," *Eng. Appl. Artif. Intell.*, vol. 120, Apr. 2023, Art. no. 105863, doi: 10.1016/j.engappai.2023.105863.

[69] U. Bhimavarapu and G. Battineni, "Deep learning for the detection and classification of diabetic retinopathy with an improved activation function," *Healthcare*, vol. 11, no. 1, p. 97, Dec. 2022, doi: 10.3390/healthcare11010097.

[70] D. Sussillo and L. F. Abbott, "Random walk initialization for training very deep feedforward networks," 2014, *arXiv:1412.6558*.

[71] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[72] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.

HAN-SHEN FENG was born in Taipei City. He received the M.S. degree from the Graduate Institute of Automation and Control, National Taiwan University of Science and Technology, Taipei City, in 2019. He is currently pursuing the Ph.D. degree with the National Taiwan University of Science and Technology. His research interests include intelligent systems, computer vision, and deep learning.

CHENG-HSIUNG YANG received the bachelor's degree from the Department of Mechanical Engineering, National Pingtung University of Science and Technology, Pingtung, Taiwan, in 1995, and the master's and Ph.D. degrees from the Department of Mechanical Engineering, National Chiao Tung University, Hsinchu, Taiwan, in 1997 and 2008, respectively. Since 2020, he has been a Professor with the Graduate Institute of Automation and Control, National Taiwan University of Science and Technology. His main research interests include chaotic encryption, FPGA system image/video processing, and computer vision.