electronics

Article

Smish: A Novel Activation Function for Deep Learning Methods

Xueliang Wang [1,2], Honge Ren [1,3,*] and Achuan Wang [1]

[1] College of Information and Computer Engineering, Northeast Forestry University, Harbin 150040, China; wangxueliang@qqhru.edu.cn (X.W.); wangca1964@126.com (A.W.)
[2] Network Information Center, Qiqihar University, Qiqihar 161006, China
[3] Forestry Intelligent Equipment Engineering Research Center, Harbin 150040, China
* Correspondence: nefu_rhe@163.com

Abstract: Activation functions are crucial in deep learning networks, given that the nonlinear ability of activation functions endows deep neural networks with real artificial intelligence. Nonlinear nonmonotonic activation functions, such as rectified linear units, Tan hyperbolic (tanh), Sigmoid, Swish, Mish, and Logish, perform well in deep learning models; however, only a few of them are widely used in mostly all applications due to their existing inconsistencies. Inspired by the MB-C-BSIF method, this study proposes Smish, a novel nonlinear activation function, expressed as $f(x) = x{\cdot}tanh[\ln(1 + sigmoid(x))]$, which could overcome other activation functions with good properties. Logarithmic operations are first used to reduce the range of sigmoid(x). The value is then calculated using the tanh operator. Inputs are ultimately used to multiply the previous value, thus exhibiting negative output regularization. Experiments show that Smish tends to operate more efficiently than Logish, Mish, and other activation functions on EfficientNet models with open datasets. Moreover, we evaluated the performance of Smish in various deep learning models and the parameters of its function $f(x) = \alpha x{\cdot}tanh[\ln(1 + sigmoid(\beta x))]$, and where $\alpha = 1$ and $\beta = 1$, Smish was found to exhibit the highest accuracy. The experimental results show that with Smish, the EfficientNetB3 network exhibits a Top-1 accuracy of 84.1% on the CIFAR-10 dataset; the EfficientNetB5 network has a Top-1 accuracy of 99.89% on the MNIST dataset; and the EfficientnetB7 network has a Top-1 accuracy of 91.14% on the SVHN dataset. These values are superior to those obtained using other state-of-the-art activation functions, which shows that Smish is more suitable for complex deep learning models.

Keywords: activation function; deep learning; image classification

Citation: Wang, X.; Ren, H.; Wang, A. Smish: A Novel Activation Function for Deep Learning Methods. *Electronics* **2022**, *11*, 540. https://doi.org/10.3390/electronics11040540

Academic Editor: Abdeldjalil Ouahabi

Received: 22 January 2022
Accepted: 7 February 2022
Published: 11 February 2022

1. Introduction

The principle of deep learning networks is that input is passed from one neuron to the next via an activation function, and the process is repeated until the output layer is reached. By repeating the aforementioned process, a linear relationship is transformed into a nonlinear relationship by using the activation function. Thus, the activation function provides deep learning networks with sufficient power to derive complex calculations and obtain state-of-the-art results in various fields. An activation function is used to improve representation ability via nonlinear operations in convolutional neural networks (CNNs). If an inappropriate activation function is selected, the residual is considerably smaller after multilayer convolutions in CNNs, requiring the termination of the training process [1]. Currently, activation functions are mainly divided into linear activation functions and nonlinear activation functions. Nonlinear activation functions can be further subdivided into the nonlinear monotonic and nonlinear nonmonotonic types [2]. The former mainly consists of simple linear and threshold functions. These functions include the step, sign, and identity functions, which are the common early linear activation functions [3–6]. The most typical and simplest linear function is the activation function [7]. However, if the output of the node is equal to the input, the activation function has no classification ability.

Linear activation functions are not widely applied in CNNs because of the discontinuous characteristics of their derivatives. In deep learning networks, nonlinear activation functions—Sigmoid, ReLU, Swish, Mish, and Logish—are frequently used [8,9]. Sigmoid maps all values to (0, 1), which is associated with the vanishing gradient problem. To address this concern, the tanh activation function is proposed [10], although it does not eliminate the aforementioned problem in deep neural networks. The activation function SoftPlus provides greater enhancement, allowing the ReLU function to further boost deep learning networks, hence its wide application used in different fields [11–14]. ReLU exhibits reduced saturation, sparsity, efficiency, and ease of use and thus can alleviate the vanishing gradient problem. However, its negative value directly returns to zero; consequently, the network loses a certain amount of valid information. Such a disadvantage has prompted the need to develop new methods, including the Hard Sigmoid and Hard Tanh activation functions, which can easily perform calculation tasks and effectively realize network learning. Meanwhile, Leaky ReLU assigns a small slope of 0.01 to negative input, and ELU contains a negative exponential term. With these approaches, the presence of silent neurons is reduced, and negative gradient learning is enabled [15,16].

In addition to nonlinear monotonic activation functions, nonlinear nonmonotonic activation functions also perform efficiently in deep learning networks. Nonlinear nonmonotonic functions, including Swish and Mish, are broadly used. Swish has an advantage in big data and deeper complex networks and thus performs more efficiently than others [17]. Proposed by Misra et al., Mish is also a nonmonotonic differentiable function, which has a lower bound but no upper bound [18]. Mish is almost smooth at any point on the curve, allowing the transformation of more valid information into the model to improve accuracy and generalization performance. In 2021, Hegui Zhu et al. proposed Logish activation, which is similar to Swish but exhibits better performance and Top-1 accuracy [2]. Proposed by Jaafari et al., PRenu activation function uses hidden layer incorporating with a chosen activation such as Relu [19]. AIS activation proposed by Wang Z et al. disposes positive and negative values separately, and its curve distribution is not very different from Mish [20]. In the data processing of the MB-C-BSIF method proposed by Adjabi et al., the input is normalized firstly and then nonlinear filtered, which makes it ideal for deep learning methods [21].

Although great success has been made by the above activation functions in some deep learning networks, the classification results are unsatisfactory in more complex networks in our study. In order to improve the accuracy rate of classification results, we designed a new activation function named Smish to solve the previously mentioned problems in deep learning networks, the aforementioned characteristics being present not only to ensure negative activation and derivative values but also to maintain partial sparsity and a regularization effect for negative inputs. Similar to Mish and Logish, Smish is a differentiable function with a lower bound and no upper bound, expressed as

$$f(x) = x \cdot tanh[\ln(1 + sigmoid(x))], \tag{1}$$

Its variant is given by

$$f(x) = \alpha x \cdot tanh[\ln(1 + sigmoid(\beta x))], \tag{2}$$

where the logarithmic operation is used to reduce the range of the sigmoid(x) function value and is calculated with the tanh function, ensuring the smoothness and stability of the normalized curve. Meanwhile, inspired by Mish, x is multiplied with the value that renders the negative output regularized, and the positive output is translated into simple linearization. The results also confirm that Smish is more suitable than Mish and Logish for learning complex networks. The innovations provided by this study are as follows:

1. A new monotonic nonlinear activation function named Smish is proposed.
2. Smish provides a higher learning accuracy, compared with Logish, Mish, Swish, and ReLu, all of which are used in several EfficientNet models.

3. Smish performs better than other functions in classification in several open datasets.

Other parts of the paper are organized as follows: Section 2 briefly introduces Mish and Logish. Section 3 presents the Smish activation function and illustrates its major properties. In Section 4, we analyze the performance of the Smish nonmonotonic functions with parameter settings. In Section 5, the Smish algorithm is used in image classification, and other common activation functions on EfficientnetBx are employed to evaluate performance. Section 6 summarizes the main results.

2. Related Work

2.1. Mish

A new activation function referred to as Mish has recently been widely used in deep-learning methods and is described as

$$f(x) = x \cdot tanh[\ln(1 + e^x)], \tag{3}$$

The derivative formula of Mish is represented as

$$f'(x) = \frac{2}{\frac{1}{(e^x+1)^2} + 1} + \frac{4xe^x}{(e^x+1)^3\left(\frac{1}{(e^x+1)^2} + 1\right)^2} - 1, \tag{4}$$

As shown in Figure 1, the Mish curve has no upper bound. Thus, positive values for Mish functions can reach any height similar to that of Swish. Theoretically, in ReLU, since the gradient of this function is zero when it is negative, "necrosis" may occur during training, and the derivative with a small negative value produces a better gradient flow than that of the derivative without response. If the input is reduced, information may be lost, rendering the activation value randomly dependent on the input. From 2013 to 2021, studies were conducted to improve an activation function for the occurrence of "necrosis" caused by ReLu. Lu et al. proposed the use of randomized asymmetric weight initialization, leaky ReLus, ELU, and so on [22,23], which allow learning the negative slope, in contrast to leaky ReLU and tanh-ReLu [24,25].

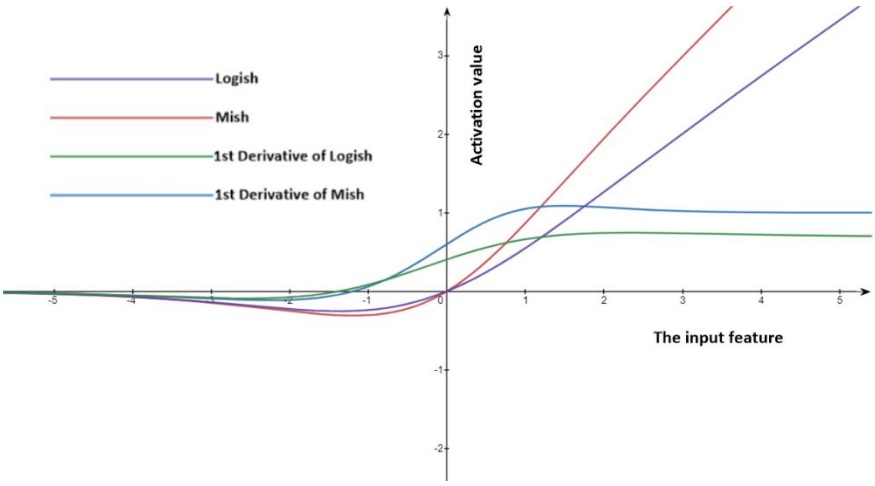

Figure 1. Curves of Mish and Logish functions and their derivatives.

Dan Hendrycks et al. introduced the concept of random regularity into the Gaussian error linear unit (GELU). It is similar to Dropout, which generates output via random multiplication by 0 or 1 [26]. Mish exhibits a random pattern similar to that of GeLU and has the advantages of being unsaturated and nonmonotonic and can easily replace ReLU. All the characteristics provide the Swish activation function with excellent response normalization. On FastAI open datasets, Mish combined with a Ranger optimizer achieves

enhanced accuracy [27]. In Mish, the derivative of a continuous order is infinite, which is an advantage over ReLU. The continuous order of ReLU is 0, indicating that it is not continuously differentiable, which can potentially present problems in gradient-based optimization.

2.2. Logish

Logish activation function is described by

$$f(x) = x \cdot \ln(1 + sigmoid(x)), \tag{5}$$

$$f'(x) = \ln\left(\frac{1}{e^{-x}+1} + 1\right) + \frac{x \cdot e^{-x}}{(e^{-x}+1)^2 \cdot \left(\frac{1}{e^{-x}+1} + 1\right)}, \tag{6}$$

As shown in Figure 1, the curve of Logish closely resembles that of Mish; no upper limit to positive values exists. Theoretically, a negative value near 0 produces a continuous differentiable derivative flow that can maintain partial sparsity. A logarithmic operation is then used in the sigmoid function to reduce the range of values and generate smooth and stable curves. The negative values are normalized, and the positive values tend to be linear. As X decreases, the input is more likely to be abandoned. Thus, in Logish, the uncertainty is preserved by deleting parts of the input value, thus increasing its dependence on other inputs [2]. Misra concluded that the smoother the activation function, the greater the ability to efficiently acquire information, the higher the network accuracy, and the better the generalization ability [11].

Additionally, for the purpose of improving the accuracy and deep learning convergence speed, the PRenu activation function uses hidden-layer incorporating with a chosen activation, which is nearly similar to Relu. However, the difference between them was that PRenu multiplies a value with a received gradient for all positive values in its back-propagation [19]. In the field of industry engineering, in the task of that the fitted model can be applied to, AIS activation consists of positive and negative regions. In the former region, the exponential and quadratic function are combined to alleviate the problem of deviation, and the linear function is added to make the positive region smoother and overcomes the problem of gradient vanishing. In the negative region, the cubic function structure is applied to solve the negative region problem and accelerate the convergence of the deep learning networks [20].

However, in complex deep learning networks, the above activation functions could not overcome the problems of gradient vanishing in negative regions, and make the deviation of networks increased, slow down the network fitting and make the feature extraction ability of the model decreased. Inspired by the MB-C-BSIF method, in the data processing stage of which the data is normalized firstly and then filtered by coefficients, Smish, which is proposed, could basically solve the above problems, and the characteristics of Smish could meet the required features [21]. The conclusions drawn from the aforementioned studies are presented as follows:

1. Mish and Logish are close to the linear function $f(x) = \alpha x$ when $x > 0$ but close to 0 when $x < 0$. These characteristics render the network more stable and directional propagation calculation easier.

2. The derivatives when $x < 0$ are not always 0, which effectively avoids gradient death.

3. When $x > 0$, the structure is similar to that of Mish; thus, the neural network exhibits sparsity and reduces computational complexity.

3. Proposed Smish

3.1. Construction of Smish

This study proposes a novel activation function called Smish, which inherits the nonmonotonic properties of the Logish function. The function is particularly suitable for deep learning networks.

$$f(x) = x{\cdot}tanh[\ln(1 + sigmoid(x))], \tag{7}$$

$$f'(x) = \frac{e^x{\cdot}\left(15{\cdot}e^{3x} + (8x+28)e^{2x} + (12x+18)e^x + 4x + 4\right)}{\left(5e^{2x} + 6e^x + 2\right)^2}, \tag{8}$$

The Smish function is consistent with the Logish function in the early stages: $sigmoid(x)$ is used to reduce the range of values, and the logarithm operation is applied to obtain a smooth curve and a flat trend [9,12]. Smish then multiplies its *tanh* operation by x simultaneously, thus exhibiting the regularization ability for negative inputs; by contrast, the positive values tend to become a simpler linear expression. The Smish curve itself and its derivatives are shown in Figure 2.

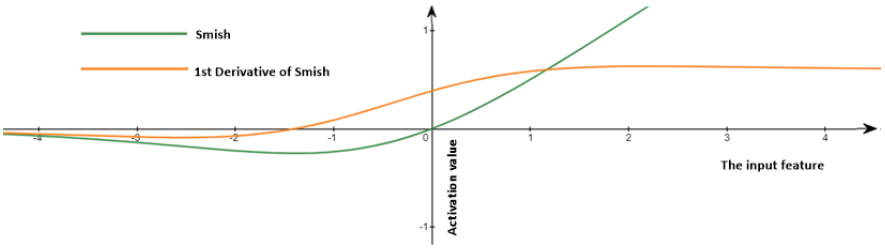

Figure 2. Curves of the Smish activation function and its derivative.

Similar to Mish and Logish, Smish has a lower bound and no upper bound, and its range is $[-0.25, +\infty)$. The minimum value of Smish is approximately -0.25, and the input value is -1.3945. It is prompted by self-gating in the long short-term memory of deep recurrent convolution, the advantage of this pattern is its convenience of replacing activation functions such as ReLU because of the transformation of input values [28]. When $x < -12.427$, for a negative value of x, the value is as same as ReLU. Smish activation functions can be easily implemented using some frameworks by customizing an activation layer. For example, in the framework of TensorFlow, it is Keras $x * K.Tanh\ (K.log(1 + K.sigmoid(x)))$ while in Torch it is $x * torch.tanh(torch.log(1 + torch.sigmoid(x)))$; it is more suitable to set the learning rate as le-4 in experiments. Formulas (7) and (8) and Figures 1 and 2 indicate that the simplified formula for Smish and Mish is exponential. As x approaches infinity, Mish is closer to x; by comparison, Smish is closer to $\frac{2^2-1}{2^2+1}x$.

3.2. Approximate Linear Transformation

As the depth of the network model increases, the value of the output does not change significantly with that of the nearly linear activation function. Owing to the nearly linear transformation characteristics of the network, it is stable and convenient for gradient backpropagation. Logish is then proved to be approximately $0.6x$.

$$f(x) = x{\cdot}\left(\frac{2}{1 + e^{-2{\cdot}\ln\left(1+\frac{1}{1+e^{-x}}\right)}} - 1\right) = x{\cdot}\frac{\left(1+\frac{1}{1+e^{-x}}\right)^2 - 1}{\left(1+\frac{1}{1+e^{-x}}\right)^2 + 1} = \frac{3}{5}x = 0.6x, \tag{9}$$

Smish reduces computational complexity by using the aforementioned calculation. The derivative of Smish for a positive x is equal to $0.6x$. Whereas the derivatives of Mish and ReLU tend to be equal to 1, Smish is lower than Logish and thus exhibits a more

gradual growth trend, compared with other functions. Thus, Smish performs better in deep learning fields.

3.3. Nonmonotonicity

A good activation function should not induce the disappearance of the gradient, and a small number of negative values are allowed to exert regularization effects [29]. In Figure 2, Smish satisfies all properties of good activation functions. Therefore, Smish is a good choice for deep learning networks. Moreover, the nonmonotonicity of Smish ensures the stability of negative training and improves the performance for expression. On the basis of Cauchy's theorem, given that $f\ (-2.5) = f\ (-0.63)$, there must be x, $x \in [-2.5, -0.63]$, $f\prime(x) = 0, x = -1.3945$. Smish decreases monotonically in $(-\infty, -1.3945)$ and increases monotonically in $(-1.3945, -\infty)$. Therefore, Smish is nonmonotonic in $(-\infty, +\infty)$, which can improve the ability of network learning and gradient transformation.

4. Analysis of Hyperparameter Tuning for Smish

This section focuses on the superiority of the Smish activation function by adjusting several classical super parameters. These parameters include batch sizes, learning rates, dropout, optimizers, and regularization methods. To better reflect the performance of the activation function, the model adopts 20 convolution layers and 2 pooling layers. Moreover, EfficientNet is employed as our network model for fine-tuning parameters in the experiments. All experiments are trained and evaluated on the CIFAR10 datasets [30,31].

4.1. Analysis of the Number of Layers

To observe the effect of network layers with the Smish activation function on EfficientNet, we trained fully connected networks at different depths on MNIST datasets with 500 neurons per layer under similar conditions. Deep residual models were not used because depth only slightly affects the training result [32]. In this section, BatchNorm was employed to reduce reliance on initialization, and the dropout rate was set to 25% [33]. The network was optimized by stochastic gradient descent (SGD), and the batch size was 128, which achieved the highest accuracy, as proved in a subsequent section [34]. For a fair comparison, the same learning rate was used to train each model with 'le-4'. In the experiment, all other activation functions maintained nearly identical levels of test accuracy for the 15-layer network. With a gradual increase in the number of layers, the testing accuracy of ReLU sharply declined from 15, and that of Swish markedly decreased from 29. Regardless, Smish outperformed the other four activation functions in simple and complex networks (Figure 3).

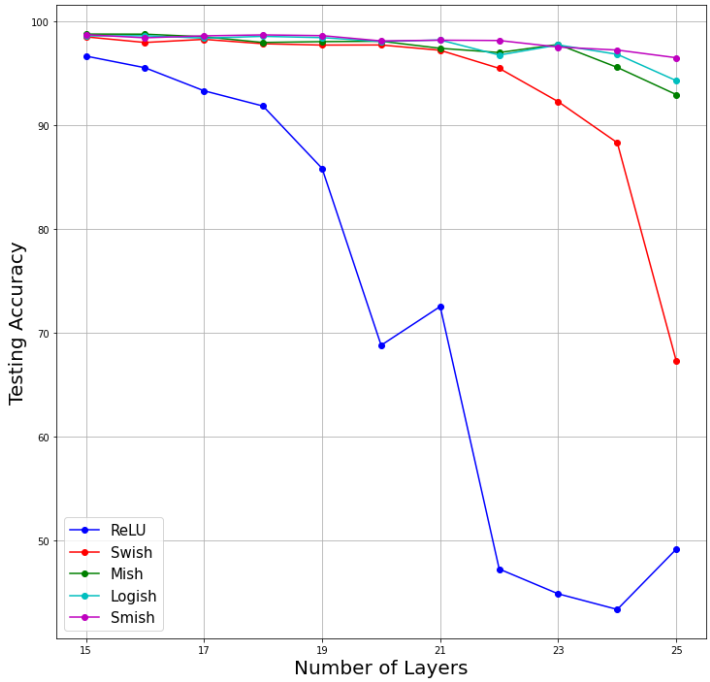

Figure 3. Testing accuracy vs. number of layers on MNIST for ReLU, Swish, Mish, Logish, and Smish.

4.2. Analysis of Batch Sizes

The batch size parameter refers to the number of training samples selected in every training epoch, which largely influences the results of model training. Generally, as batch_size increases, the gradient descent changes more accurately; however, gradients can easily disappear. By contrast, as the situation is reversed, the training time becomes longer. In our study, the conditions were set as follows: epoch, 100 on CIFAR10; learning rate, 0.0001; dropout rate, 0.8. All network parameters were kept constant, and the batch size was set to 16, 32, 64, 128, 256, 512, and 1024. In Figure 4, the highest accuracy of all five activation functions occurs when the batch size is set to 128, where Smish obtained the best performance on CIFAR10. Moreover, Smish is higher than the other activation functions during the entire training process for batch size.

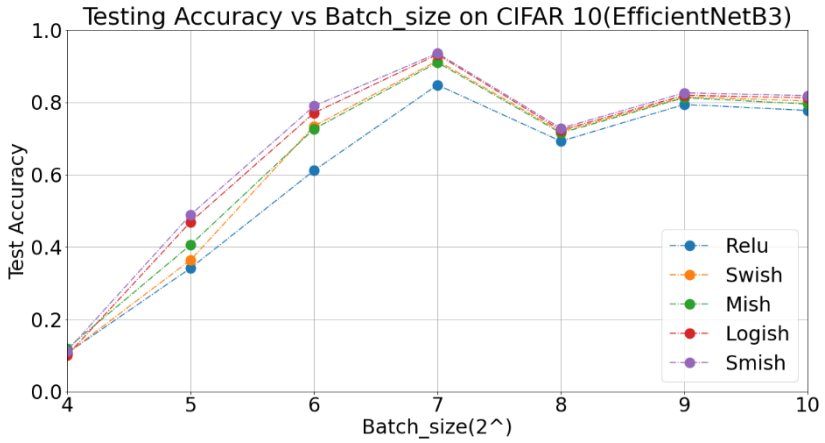

Figure 4. Training accuracy vs. batch_size on CIFAR10.

In addition, as shown in Figure 5, time is also an important factor for deep learning samples. With Smish as the activation function, when the batch size was set to 128, the least time and highest accuracy were attained; when the batch size was set to other values, more time was consumed. Thus, in subsequent experiments, batch_size was set to 128.

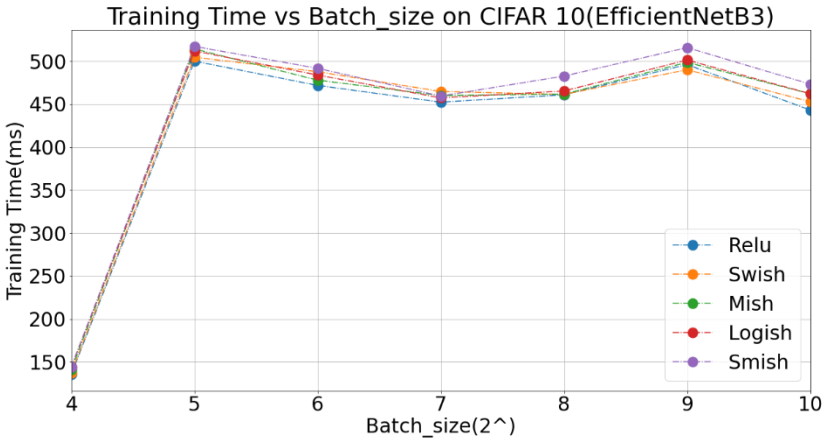

Figure 5. Training time vs. batch_size on CIFAR10.

4.3. Analysis of learning rates

The learning rate plays an important role in the optimization of deep learning networks, and the value for the learning rate is always inconclusive. The learning rate of the training sample was selected from le-1 to le-7 for testing, and the CIFAR10 dataset was trained. As shown in Figure 6, ReLU initially exhibits the highest accuracy, but overall, Smish shows the highest accuracy (0.91). Although Logish performs best when the learning rate was le-3, its performance remained lower than that of Smish. Subsequently, as the learning rate decreased, the testing accuracy levels of all functions decreased simultaneously, with ReLU exhibiting the lowest accuracy. These results indicate that Smish significantly outperformed other functions at higher learning rates. In subsequent experiments, the learning rate was set to le-4.

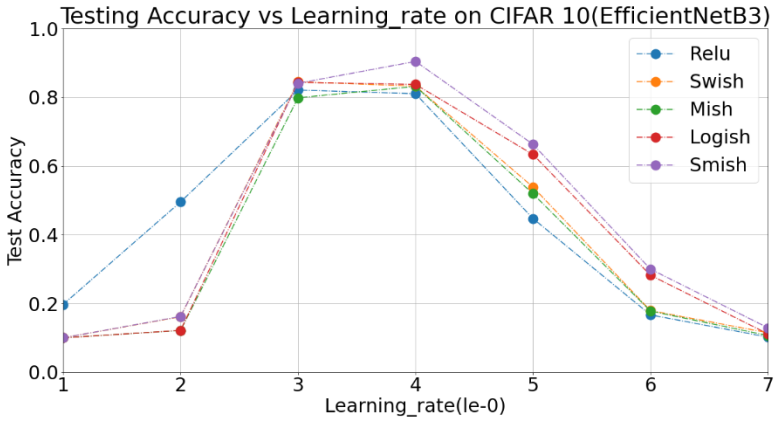

Figure 6. Training accuracy vs. learning rate.

4.4. Analysis of Different Dropout Rates

The dropout layer processes some features of the hidden layer to regularize and suppress overfitting. Nine values ranging from 0.1 to 0.9 were selected. The test accuracy levels of the five activation functions are shown in Figure 7. Overall, dropout only slightly affected the accuracy of the model. Smish performed more efficiently than other activation functions, achieving the highest accuracy at a dropout of 0.2. The synergistic effect between different features of the effective link exhibited high robustness.

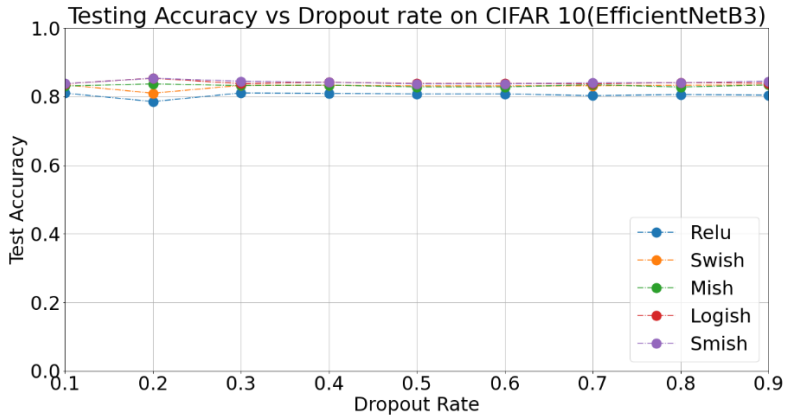

Figure 7. Test accuracy vs. dropout rate.

4.5. Analysis of Different Optimizers

In deep learning, an appropriate optimizer is selected, and the super parameters are adjusted to extract information from the input to allow it to adapt to the output value. The aim is to identify the optimal solution. Thus, choosing the appropriate optimizer is important in network training. In the current study, we selected Adadelta [35], Adagrad [36], Adam [37], RMSprop [38], and SGD [39] optimizers. As shown in Figure 8, Smish achieves an accuracy of 0.9 with Adam and 0.83 with RMSprop, which are significantly superior to the accuracy rates of the other activation functions. Logish exhibited an accuracy of 0.84 with Adam, and several other activation functions showed accuracy rates higher than 0.8 with Adam. Thus, with any optimizer, Smish performed better than any other activation function.

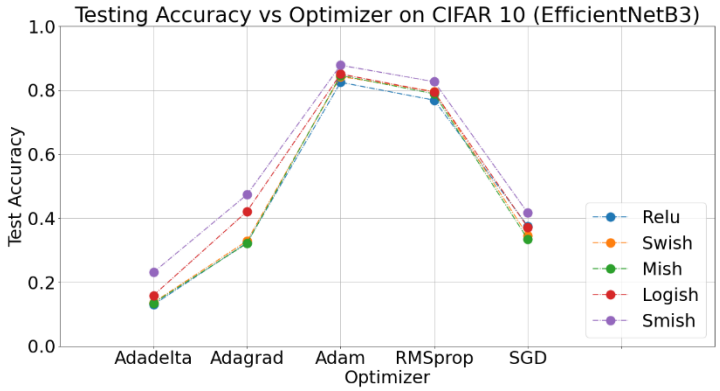

Figure 8. Test accuracy vs. optimizer.

4.6. Analysis of Different Parameters

The evaluation of the performances of Smish and other compared functions in a deep learning model is discussed in previous sections. This section verifies the influence of α and β parameters in $f(x) = \alpha x \cdot tanh[\ln(1 + sigmoid(\beta x))]$ on the experimental model. Table 1 shows the training time of the selected model and the loss value of the test samples and verification accuracy of different values of α and β.

Table 1. Training time, accuracy, and loss with the Smish activation function at different values of α and β.

α	β	Train Time (ms)	Loss	Val_Accuracy
20	1	1664.87985	2.093894768	0.317399998
15	1	1657.337511	2.084924269	0.326699999
10	1	1688.491895	2.054765987	0.356000003
5	1	1683.588445	1.366328526	0.626699996
0.1	1	1697.313161	1.278249788	0.790999961
0.01	1	1719.354016	1.111771631	0.818499994
0.005	1	1708.812285	2.102614927	0.299999994
0.001	1	1718.249439	2.102625656	0.299999994
1	20	1722.055284	1.059448051	0.717399988
1	15	1723.015788	1.032156754	0.723999621
1	10	1717.969835	1.057105231	0.755299997
1	5	1727.269962	1.081895041	0.851999986
1	1	1724.575549	0.870445440	0.909814782
1	0.5	1717.444711	1.033060956	0.833099973
1	0.1	1725.162607	0.92743938	0.815499973
1	0.05	1730.499959	1.000445652	0.791300011
1	0.01	1731.330632	1.108899283	0.755499971

As shown in Table 1, when $\beta = 1$ and $\alpha > 1$, the smaller the value of α, the higher the accuracy. When $0 < \alpha < 1$, the accuracy decreases with a decrease in α. When $\alpha = 1$ and $\beta > 1$, the accuracy was inversely proportional with the increase in β. When $0 < \beta < 1$, the smaller the value of β, the lower the accuracy. The highest accuracy was achieved when both α and β were set to one; thus, in subsequent experiments, both α and β were assigned to one.

5. Experimental Results of Open Datasets

In this chapter, the performance of Smish relative to other activation functions is evaluated with three open datasets and several EfficientNet models. Notably, our evaluation is based on test samples rather than training samples.

5.1. Datasets and Experimental Settings

All models were trained and tested on MNIST, CIFAR10, and SVHN datasets. MNIST datasets contained 60,000 training samples and 10,000 test samples, which were sorted into 10 categories—that is, "0–9", ten Arabic numerals. The CIFAR10 dataset contained 50,000 training samples and 10,000 test samples, and these photos were RGB data images with three $32 * 32$ channels. The SVHN datasets were derived from the Street View House Number (SVHN) dataset, and each image contained a set of Arabic numerals '0–9'. As shown in the image below, the training samples contain 73,257 digits, 26,032 test digits, and 531,131 additional digits. All experiments were performed on TensorFlow GPU using Python 3.7 with the following parameters: batch_size, 128; learning_rate, 0.0001; epoch, 100; and optimizer, Adam. The training process is as follows:

First, we determined from the experiments that data enhancement exerted no considerable effect on the training results of several models but increased computational cost. Thus, in the experiment, we did not enhance data for initialization. Second, we chose four EfficientNet models, considering that EfficientNet has been proved in numerous literature reviews to have a greater advantage in image deep learning, compared with DensenNet, ResNet [18], EfficientNetB3, EfficientNetB5, EfficientNetB6, and EfficientNetB7 [40]. EfficientNetB0 to EfficientNetB7 included seven blocks. The most critical aspect of any network was its stem, which was determined before subsequent experiments were conducted. This structure was common in all eight models and the last layer. The blocks also varied in the number of subblocks, which could increase as we moved from EfficientNetB0

to EfficientNetB7. The differences between all models gradually increased the number of subblocks.

5.2. Classification Result on CIFAR10

The classification results on CIFAR10, obtained via the activation functions ReLU, Swish, Mish, Logish, and Smish, are listed in Table 2. The first four functions were trained on EfficientNetB3, B5, B6, and B7 networks, respectively. We could conclude that Smish consistently outperformed the other four activation functions in any model on CIFAR10. In EfficientNetB3, Smish achieved an improvement of 0.01 relative to Logish and 0.018 relative to Mish. The improvement in EfficientNetB5 and EfficientNetB7 was not significant, but Smish outperformed the others. The Smish classification accuracy of 0.85 in EfficientNetb5–7 proved that it could be widely used in complex models, compared with other activation functions. As indicated in Figures 9 and 10, Smish had the lowest value, compared with the other activation functions, and consistently attained the highest accuracy in any epoch.

Table 2. Accuracy of EfficientNet models achieved via the five activation functions on CIFAR10.

Model	EfficientNetB3	EfficientNetB5	EfficientNetB6	EfficientNetB7
ReLU	0.812400	0.818400	0.838900	0.837200
Swish	0.820700	0.840400	0.846900	0.856600
Mish	0.823200	0.837800	0.842700	0.853000
Logish	0.833500	0.843500	0.850700	0.858000
Smish	0.841000	0.854900	0.851000	0.859400

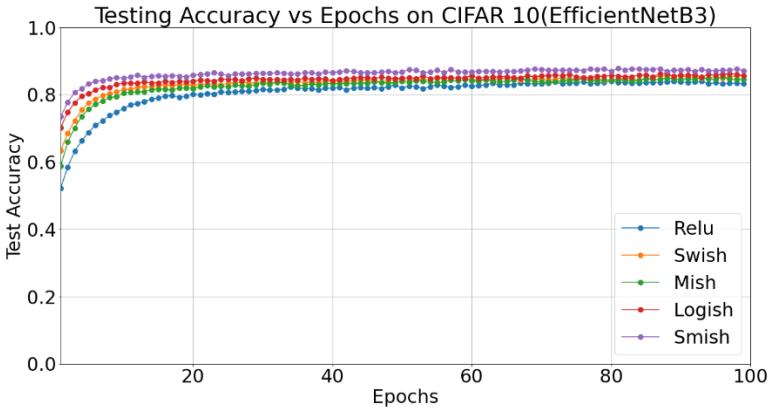

Figure 9. Accuracy vs. epoch in EfficientNetB3 achieved via the five activation functions on CIFAR10.

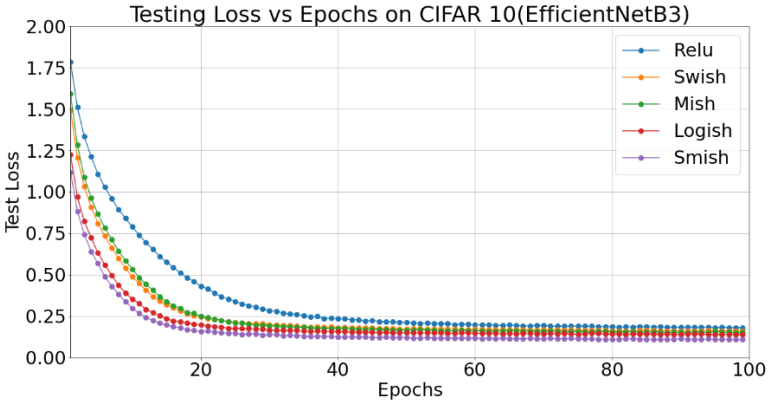

Figure 10. Loss vs. epochs in EfficientNetB3 achieved via the five activation functions on CIFAR10.

5.3. Classification Results on MNIST

We also conducted experiments on MNIST datasets to evaluate the performance of several activation functions. As shown in Table 3, Smish performs better than the other four activation functions. In EfficientNetB3, Smish improved by 0.05 relative to Logish and by 0.5 relative to Mish. The EfficientNetB6 and EfficientNetB7 results on CIFAR10 were similar. Thus, for more complex models, an increase in parameters was not markedly effective; however, the effect was better on Smish than on the four other activation functions, as determined from its accuracy of 0.99. Smish also consistently outperformed the other four activation functions in loss and accuracy epochs, respectively (Figures 11 and 12).

Table 3. Accuracy of EfficientNet models achieved via the five activation functions on Mnist.

Model	EfficientNetB3	EfficientNetB5	EfficientNetB6	EfficientNetB7
ReLu	0.564600	0.128999	0.986900	0.990800
Swish	0.912500	0.752599	0.987700	0.991900
Mish	0.397300	0.189899	0.987800	0.992600
Logish	0.925700	0.784300	0.990100	0.992800
Smish	0.977800	0.966500	0.998900	0.993500

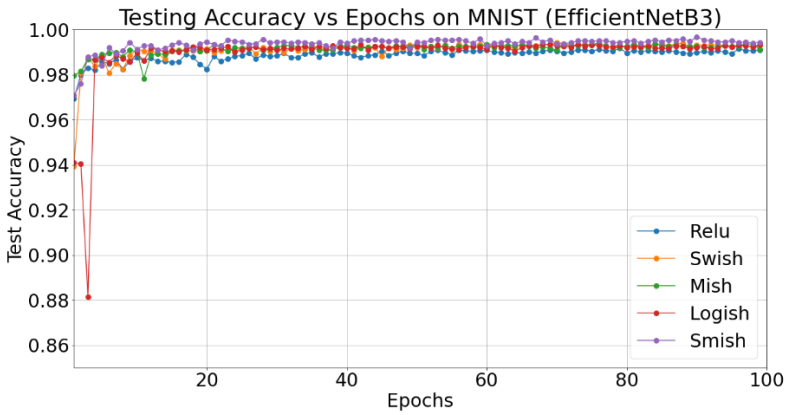

Figure 11. Accuracy vs. epochs in EfficientNetB3 achieved via the five activation functions on Mnist.

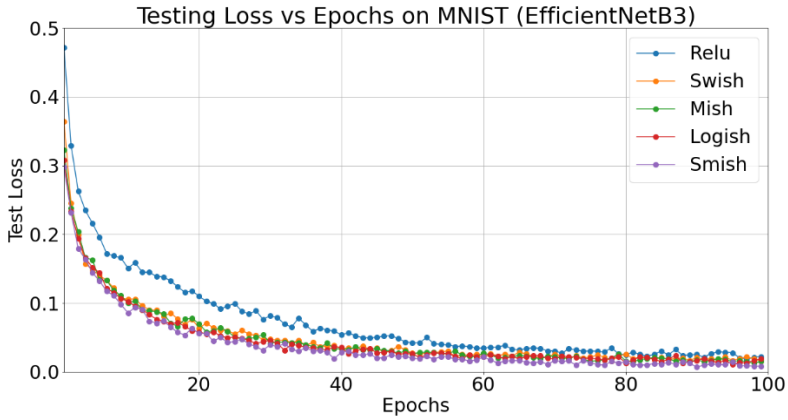

Figure 12. Loss vs. epochs in EfficientNetB3 achieved via the five activation functions on Mnist.

5.4. Classification Results on SVHN

The experimental accuracy levels of the five functions on the SVHN dataset were ultimately compared. Among the four EfficientNet models, Smish achieved the largest improvement in classification (Table 4). In EfficientNetB3, Smish attained an improvement of 0.05 relative to ReLU and 0.05 relative to Logish. In EfficientNetB7, Smish improved in

accuracy by 0.06 relative to Logish, 0.7 relative to ReLU and Swish, and 0.2 relative to Mish. Smish outperformed the other activation functions (Figures 13 and 14).

Table 4. Accuracy of EfficientNet models achieved via the five activation functions on SVHN.

Model	EfficientNetB3	EfficientNetB5	EfficientNetB6	EfficientNetB7
ReLu	0.829095	0.854179	0.851605	0.195874
Swish	0.860787	0.882683	0.844729	0.195874
Mish	0.864897	0.889405	0.830670	0.775738
Logish	0.863745	0.890174	0.899739	0.848494
Smish	0.874232	0.893977	0.901467	0.911417

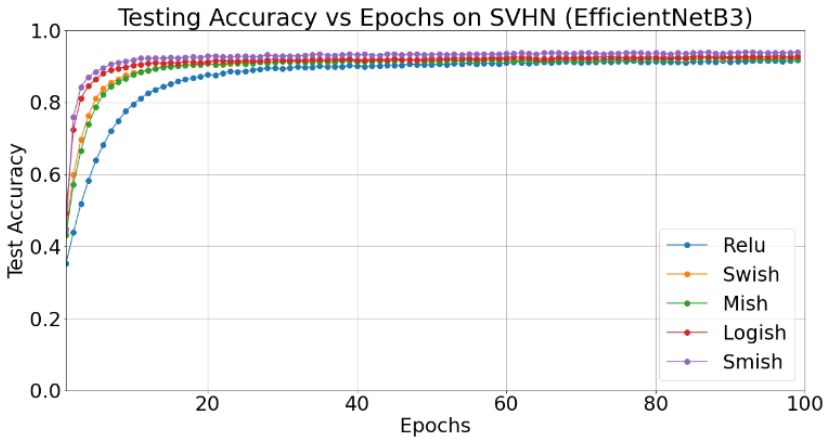

Figure 13. Accuracy vs. epochs in EfficientNetB3 achieved via the five activation functions on SVHN.

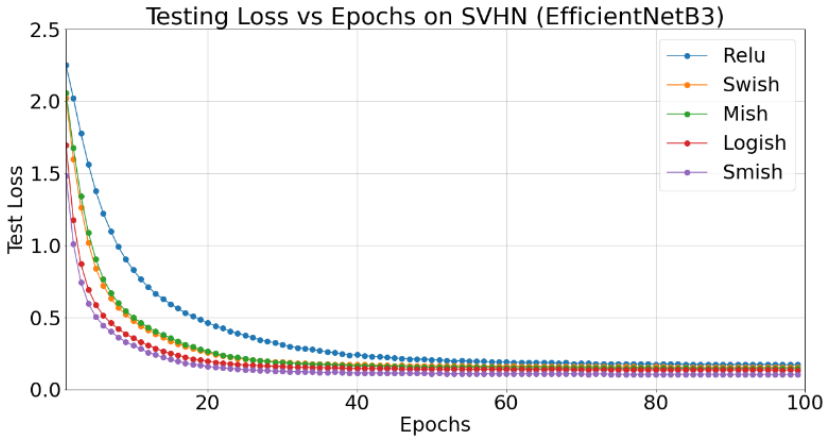

Figure 14. Loss vs. epochs in EfficientNetB3 achieved via the five activation functions on SVHN.

6. Conclusions

In our study, we proposed Smish, a deep learning activation function and its variation to improve the classification performance of deep learning models. Smish is expressed as $f(x) = x \cdot tanh[\ln(1 + sigmoid(x))]$, which is a smooth nonmonotonic function with a lower bound but without an upper bound. These properties of Smish enable deep learning networks to benefit negative representation, which achieves better performance. Simulations and experiments confirmed the effectiveness of Smish, which outperformed ReLu, Swish, Mish, and Logish in EfficientNetBx. Smish also performed more efficiently when learning rates were set to higher values. Therefore, Smish is a better alternative to ReLu than Mish and Logish.

Obviously, the complexity of Smish is higher than other compared activation functions, so there are some limitations in the lightweight models, which will be solved in the future

research. Probably the most improvement of this study would be the consideration of restricting calculation by subdividing the interval in different conditions so as to expand its scope of applications [41].

Author Contributions: X.W. proposed the algorithm and performed the experiment. H.R. supervised the study, analyzed the results, and provided insightful suggestions for the manuscript. A.W. drafted the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Fundamental Research Funds for the Central Universities (No. 2572017PZ10) and Basic Scientific Research Project of Heilongjiang Provincial Universities (No. 145109219). All the works were conducted at the Forestry Intelligent Equipment Engineering Research Center.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to still being used in a proceeding project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *6*, 84–90. [CrossRef]
2. Zhu, H.; Zeng, H.; Liu, J.; Zhang, X. Logish: A new nonlinear nonmonotonic activation function for convolutional neural network. *Neurocomputing* **2021**, *458*, 490–499. [CrossRef]
3. Hayou, S.; Doucet, A.; Rousseau, J. On the impact of the activation function on deep neural networks training. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 2672–2680.
4. Dureja, A.; Pahwa, P. Analysis of non-linear activation functions for classification tasks using convolutional neural networks. *Recent Pat. Comput. Sci.* **2019**, *12*, 156–161. [CrossRef]
5. Hu, X.; Liu, W.; Bian, J.; Pei, J. Measuring model complexity of neural networks with curve activation functions. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 6–10 July 2020; pp. 1521–1531.
6. Obla, S.; Gong, X.; Aloufi, A.; Hu, P.; Takabi, D. Effective activation functions for homomorphic evaluation of deep neural networks. *IEEE Access* **2020**, *8*, 153098–153112. [CrossRef]
7. Maguolo, G.; Nanni, L.; Ghidoni, S. Ensemble of convolutional neural networks trained with different activation functions. *Expert Syst. Appl.* **2021**, *166*, 114048. [CrossRef]
8. Zhu, M.; Min, W.; Wang, Q.; Zou, S.; Chen, X. PFLU and FPFLU: Two novel non-monotonic activation functions in convolutional neural networks. *Neurocomputing* **2021**, *429*, 110–117. [CrossRef]
9. Frasin, B.A.; Swamy, S.R.; Nirmala, J. Some special families of holomorphic and Al-Oboudi type bi-univalent functions related to k-Fibonacci numbers involving modified Sigmoid activation function. *Afr. Mat.* **2021**, *32*, 631–643. [CrossRef]
10. Goyal, M.; Goyal, R.; Lall, B. Learning Activation Functions: A new paradigm for understanding Neural Networks. *arXiv* **2019**, arXiv:1906.09529.
11. Misra, D. Mish: A self-regularized non-monotonic neural activation function. *arXiv* **2019**, arXiv:1908.08681.
12. Chiluveru, S.R.; Chunarkar, S.; Tripathy, M.; Kaushik, B.K. Efficient Hardware Implementation of DNN-based Speech Enhancement Algorithm with Precise Sigmoid Activation Function. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 3461–3465. [CrossRef]
13. Niu, X.F.; Ma, W.P. A novel quantum neural network based on multi-level activation function. *Laser Phys. Lett.* **2021**, *18*, 025201. [CrossRef]
14. Pomogaev, A.S.; Dementev, D.A.; Krasnenko, D.M.; Shtylenko, A.S. Exploring the possibility of applying different neuronal activation functions to a single-circuit. *ACS J. Phys. Conf. Series* **2021**, *1889*, 022007. [CrossRef]
15. Wuraola, A.; Patel, N.; Nguang, S.K. Efficient activation functions for embedded inference engines. *Neurocomputing* **2021**, *442*, 73–88. [CrossRef]
16. Parisi, L.; Neagu, D.; Ma, R.; Campean, F. Quantum ReLU activation for Convolutional Neural Networks to improve diagnosis of Parkinson's disease and COVID-19. *Expert Syst. Appl.* **2022**, *187*, 115892. [CrossRef]
17. Gulcehre, C.; Moczulski, M.; Denil, M.; Bengio, Y. Noisy activation functions. In Proceedings of the International conference on machine learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 3059–3068.
18. Cheridito, P.; Jentzen, A.; Riekert, A.; Rossmannek, F. A proof of convergence for gradient descent in the training of artificial neural networks for constant target functions. *J. Complex.* **2022**, 101646, (*in press*). [CrossRef]
19. Rammo, F.M.; Al-Hamdani, M.N. Detecting the Speaker Language Using CNN Deep Learning Algorithm. *Iraqi J. Comput. Sci. Math.* **2022**, *3*, 43–52. [CrossRef]

20. Kapatsinski, V. Learning fast while avoiding spurious excitement and overcoming cue competition requires setting unachievable goals: Reasons for using the logistic activation function in learning to predict categorical outcomes. *Lang. Cogn. Neurosci.* **2021**, 1–22. [CrossRef]
21. Adjabi, I.; Ouahabi, A.; Benzaoui, A.; Jacques, S. Multi-block color-binarized statistical images for single-sample face recognition. *Sensors* **2021**, *21*, 728. [CrossRef]
22. Apicella, A.; Donnarumma, F.; Isgrò, F.; Prevete, R. A survey on modern trainable activation functions. *Neural Netw.* **2021**, *138*, 14–32. [CrossRef]
23. Zuo, Z.; Li, J.; Wei, B.; Yang, L.; Fei, C.; Naik, N. Adaptive Activation Function Generation Through Fuzzy Inference for Grooming Text Categorisation. In Proceedings of the 2019 IEEE International Conference on Fuzzy Systems, New Orleans, LA, USA, 23–26 June 2019.
24. Tsai, Y.H.; Jheng, Y.J.; Tsaih, R.H. The Cramming, Softening and Integrating Learning Algorithm with Parametric ReLu Activation Function for Binary Input/Output Problems. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–7, 30.
25. Yang, J.; Duan, A.; Li, K.; Yin, Z. Prediction of vehicle casualties in major traffic accidents based on neural network. In *AIP Conference Proceedings*; AIP Publishing: Chongqing, China, 2019; Volume 2073, p. 020098.
26. El Jaafari, I.; Ellahyani, A.; Charfi, S. Parametric rectified nonlinear unit (PRenu) for convolution neural networks. *Signal Image Video Processing* **2021**, *15*, 241–246. [CrossRef]
27. Wang, Z.; Zhang, B.; Gao, D. AIS: A nonlinear activation function for industrial safety engineering. *arXiv* **2021**, arXiv:2111.13861.
28. Hendrycks, D.; Gimpel, K. Gaussian Error Linear Units (Gelus). *arXiv* **2016**, arXiv:1606.08415.
29. Duan, Q.; Li, X.; Yin, Q.; Feng, L.; Zhao, J.; Teng, Y.; Duan, X.; Zhao, Y.; Gao, M.; Wang, J.; et al. A Study on the Generalized Normalization Transformation Activation Function in Deep Learning Based Image Compression. In Proceedings of the 6th International Congress on Information and Communication Technology; Springer: Singapore, 2022; pp. 351–359.
30. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
31. Abouelnaga, Y.; Ali, O.S.; Rady, H.; Moustafa, M. Cifar-10: Knn-based ensemble of classifiers. In Proceedings of the 2016 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 15–17 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1192–1195.
32. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
33. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 7–9 July 2015; pp. 448–456.
34. Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the COMPSTAT'2010*; Physica-Verlag HD: Berlin/Heidelberg, Germany, 2010; pp. 177–186.
35. Zeiler, M.D. Adadelta: An adaptive learning rate method. *arXiv* **2012**, arXiv:1212.5701.
36. Lydia, A.; Francis, S. Adagrad—An optimizer for stochastic gradient descent. *Int. J. Inf. Comput. Sci.* **2019**, *6*, 566–568.
37. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
38. Bingham, G.; Miikkulainen, R. Discovering parametric activation functions. *Neural Netw.* **2022**. [CrossRef]
39. Park, J.; Kim, M.J.; Jung, W.; Ahn, J.H. AESPA: Accuracy Preserving Low-degree Polynomial Activation for Fast Private Inference. *arXiv* **2022**, arXiv:2201.06699.
40. Xie, C.; Tan, M.; Gong, B.; Wang, J.; Yuille, A.L.; Le, Q.V. Adversarial examples improve image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 819–828.
41. You, Z.; Gao, H.; Li, S.; Guo, L.; Liu, Y.; Li, J. Multiple activation functions and data augmentation based light weight network for in-situ tool condition monitoring. *IEEE Trans. Ind. Electron.* **2022**, 1. [CrossRef]