Journal of Advances in Information Technology, Vol. 17, No. 3, 2026

LSGELU: Improved Gaussian Error Linear Units for Enhanced Performance in Simple and Complex Neural Networks

Suwichai Phunsa [1] and Thawatchai Chomsiri [2,*]

[1] Department of Creative Media, Digital Contents for Development Research Unit, Faculty of Informatics, Mahasarakham University, Mahasarakham, Thailand
[2] Department of Information Technology, Research Center of Information Technology for the Future, Faculty of Informatics, Mahasarakham University, Mahasarakham, Thailand
Email: suwichai.p@msu.ac.th (S.P.); thawatchai.c@msu.ac.th (T.C.)
*Corresponding author

Abstract—**In this paper, we propose Left-Shifted Gaussian Error Linear Units (LSGELU), a novel modification of the Gaussian Error Linear Unit (GELU) in which the base Gaussian curve is horizontally shifted to produce altered activation responses. Specifically, left shifts move the Gaussian center from $x = 0$ to $x = −1.5$ (a 1.5-unit leftward displacement), while right shifts move the center 1.5 units rightward along the x-axis. By displacing the Gaussian kernel, the integral that defines the GELU activation is modified, yielding a parameterized activation function whose dip (local trough) can be adjusted to better suit network dynamics. We evaluate the proposed variants on Modified National Institute of Standards and Technology (MNIST), Street View House Numbers (SVHN), Canadian Institute for Advanced Research-10 classes (CIFAR-10), and CIFAR-100. The experiments include multiple independent runs and report mean accuracy, standard deviation, and confidence intervals (with degrees of freedom specified). The results indicate that left-shifted LSGELU variants consistently improve performance for relatively simple architectures (with fewer hidden layers and units), whereas right-shifted variants Right-Shifted Gaussian Error Linear Units (RSGELU) are more effective for deeper, more complex networks when used in convolutional neural networks of varying depth and capacity. These findings suggest that controllable horizontal shifts of the GELU kernel provide a low-cost, interpretable mechanism for adapting activation behavior to model complexity, offering a practical avenue for performance tuning in image classification tasks.**

Keywords—**Left-Shifted Gaussian Error Linear Units (LSGELU), Right-Shifted Gaussian Error Linear Units (RSGELU), activation function, neural networks, machine learning**

I. Introduction

Recently, deep learning [1] has emerged as a vital and widespread technique, applied across various fields such as image processing, natural language processing, and speech recognition. The crucial element that enhances a model's capabilities is the activation function, which plays a vital role in introducing nonlinearity, enabling the model to learn the deep-seated patterns and characteristics within data more effectively. From the introduction of Rectified Linear Unit (ReLU) [2–4], which is a simple yet powerful function for reducing the vanishing gradient problem, to the development of other functions such as Exponential Linear Unit (ELU) [5–7] and Gaussian Error Linear Unit (GELU) [8–10], which provide high performance across various tasks, the development of activation functions continues to progress steadily. This ongoing advancement responds to the demands of increasingly complex models and improvements in training efficiency. In this research, we focus on presenting a novel activation function called Left-Shifted Gaussian Error Linear Units (LSGELU), which builds upon GELU by enhancing adaptability and stability in learning. The proposed activation function is expected to boost performance and serve as an alternative for creating deep learning models across various domains.

In this study, we conducted experiments using the Modified National Institute of Standards and Technology (MNIST) dataset [11, 12], which consists of sample images of digital digits in a 28×28-pixel format. This dataset is widely used for testing and comparing the performance of image classification models in deep learning. The structure of this paper is divided into five sections. Section I provides the introduction, which corresponds to the current section. Section II presents the background and reviews several important and widely used activation functions. Section III introduces the LSGELU formulation. Section IV describes the experimental setup and discusses the results. Finally, Section V concludes and further works.

II. Background

Understanding the underlying technology of activation functions is crucial because it represents the model's ability to learn complex patterns in data. These functions

Manuscript received August 6, 2025; revised August 18, 2025; accepted November 24, 2025; published March 10, 2026.

not only create nonlinearity but also enable models to adapt and learn efficiently. In earlier eras, ReLU [2–4] revolutionized the field with its simplicity and effectiveness, helping to reduce the vanishing gradient problem and accelerate model training speed. Subsequently, other functions were developed, such as Leaky ReLU [9], ELU [5–7], and GELU [8–10], which employed statistical concepts and probability to create greater balance and accuracy. This development also supported models' ability to formulate hypotheses and adapt better across different data domains. Furthermore, many concepts in the academic community are currently focusing on creating functions with context-adaptive capabilities, including probabilistic integration that helps models operate not solely on average values but also opens opportunities for greater diversity and controllability.

Beyond activation functions like ReLU and ELU, there are currently new functions being developed that can respond even better to model complexity, such as Swish [13, 14], was developed to replace ReLU as a function that can adapt continuously and maintain balance between linearity and nonlinearity. Additionally, Mish [15] is a function that allows weights and variables to flow through smoothly, enabling models to learn complex data with greater precision. SoftPlus [16] is a function that reduces Sigmoid's drawbacks through smoother characteristics and a wider value range, helping to accelerate model training. SoftSign [17] is another simple function that gradually approaches values of 1 and −1, respectively, helping to reduce vanishing gradient problems and zero gradient occurrences that may arise in Sigmoid [18] type functions.

Sigmoid [18] and SoftMax [19], in the context of activation functions, both serve as fundamental structures for classification models, particularly in the final stages of models for various classification tasks. Sigmoid transforms data values to the range of 0 to 1, making it suitable for binary classification tasks, though it may experience vanishing gradient problems when used in deep layers. SoftMax is designed to convert values into probabilities for each class, which is used effectively in multi-class classification work, though caution must be taken regarding overconfidence in results.

III. Formulation

After studying the GELU and its graph, the research team discovered that in the region where $x > 0$, the curvature of the GELU graph exhibits a rightward bias when compared to the ReLU graph. As the value of x increases, the graph eventually curves back to approach the ReLU line more closely. This observation sparked our curiosity about whether shifting the GELU graph slightly to the left would enhance various performance metrics, such as achieving higher accuracy or obtaining lower loss values, by making the right side of the graph (the region where $x > 0$) more closely resemble the ReLU graph. Our objective was to shift the graph so that the valley moves further to the left while ensuring that the curve continues to pass through the point (0, 0) as before. This reasoning led us to develop the formula for LSGELU.

First, we introduce the ReLU equation:

$$\mathrm{ReLU}(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{1}$$

ReLU function in Eq. (1) is an activation function with two regions: for $x < 0$ it returns 0, and for $x \geq 0$ it returns x.

Second, we introduce the GELU formula in Eq. (2), since our work builds on it:

$$\begin{aligned} \mathrm{GELU}(x) &= x \times P(X \leq x) \\ &= x \times \phi(x) \\ &= x \times \tfrac{1}{2}\left(1 + \mathrm{erf}(x/\sqrt{2})\right) \end{aligned} \tag{2}$$

where ϕ denotes the standard normal cumulative distribution function and erf is the error function.

The first published research paper on GELU determined $\sigma = 1$, and $\mu = 0$, and used these values throughout the paper. From our study, we learned that $\sigma = 1$, and $\mu = 0$ are fundamental parameters of the Bell Curve (Normal Distribution Curve), with half the area under the curve lying to the left of the y-axis and the other half to the right of the y-axis. The inventors of GELU multiplied this area under the curve equation by x to obtain the GELU equation. We have modified the GELU formula by determining $\sigma = 1$, and $\mu = 0$, similar to GELU, but shifting the bell curve 1.5 units to the left, resulting in the LSGELU formula in Eq. (3):

$$\begin{aligned} \mathrm{LSGELU}(x) &= x \cdot P\big(X \leq (x + W\sigma)\big) \\ &= x \cdot \phi(x + W\sigma) \\ &= x \cdot \tfrac{1}{2}\left(1 + \mathrm{erf}((x + W\sigma)/\sqrt{2})\right) \end{aligned} \tag{3}$$

We retain $\sigma = 1$ and $\mu = 0$ as in GELU for two reasons. First, keeping σ unchanged preserves the bell curve's scale—since increasing σ broadens and flattens the curve, altering σ would change its overall shape. Second, keeping μ fixed isolates horizontal displacement to our new parameter W, so with $\sigma = 1$ and $\mu = 0$ (center at $x = 0$) the left–right position of the bell curve is controlled solely by W.

We chose a 1.5-unit left shift for LSGELU (and a 1.5-unit right shift for Right-Shifted Gaussian Error Linear Units (RSGELU)) because experimental results indicate this is an appropriate default. In our trials, larger shifts (e.g., 3.0) produced very low accuracy (near zero), so 1.5 was selected based on empirical observation. We have not yet studied in depth whether an optimal shift might be π/2 (≈1.5708), but comparative experiments using shifts of 1.5 and π/2 yielded very similar outcomes. Notably, a 1.5-unit shift changes the bell curve's area split from 50%:50% to approximately 93.32%:6.68%, and performance for this split is similar to that observed for other imbalanced splits such as 95%:5% or 90%:10%.

We let S be the shift amount where $S = W\sigma$. Therefore, the LSGELU formula for default values can be rewritten as follows:

$$\begin{aligned} \mathrm{LSGELU}(x) &= x \times P\big(X \leq (x + S)\big) \\ &= x \times \phi(x + S) \\ &= x \times \tfrac{1}{2}\left(1 + \mathrm{erf}((x + S)/\sqrt{2})\right) \end{aligned} \tag{4}$$

We set the default value of S in Eq. (4) to 1.5. Therefore, the LSGELU formula for default values can be rewritten as follows:

$$\begin{aligned}\mathrm{LSGELU}(x) &= x \cdot P\big(X \leq (x+1.5)\big)\\ &= x \cdot \Phi(x+1.5)\\ &= x \cdot \tfrac{1}{2}\left(1+\mathrm{erf}((x+1.5)/\sqrt{2})\right)\end{aligned} \tag{5}$$

When we plot the LSGELU formula from Eq. (5) above and compare it with ReLU, GELU, and ELU, we obtain the graph lines as shown in Fig. 1.

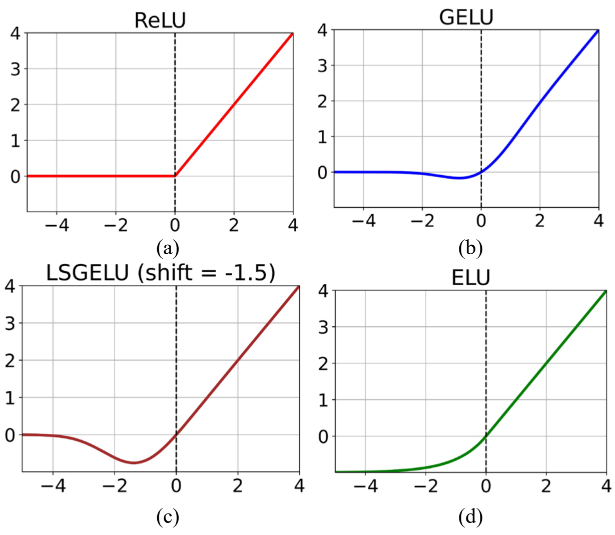


Fig. 1. Comparison of graph shapes of (a) ReLU; (b) GELU; (c) LSGELU; and (d) ELU.

From Fig. 1, when examining the right-hand side ($x > 0$), we can observe that the brown line graph of LSGELU closely resembles the ReLU and ELU graphs when compared to the GELU graphs. When considering the curvature of the GELU graph line, we can see that it curves upward to approach ReLU very closely when x reaches approximately 2.00–2.50, whereas the LSGELU graph line shows virtually no visible difference from the ReLU graph line.

When examining the left-hand side ($x < 0$), we can see that the LSGELU graph curves lower than the GELU graph. We used tools from www.desmos.com [20] to verify and found that the LSGELU graph reaches its lowest point at (−1.38136, −0.75591), while the GELU graph reaches its lowest point at (−0.75179, −0.16997). As for the ELU graph, if we start observing from point $x = 0$ and follow the graph line running toward the left, we can see that it continues to decrease and approaches the line $y = -1$, which does not curve back up toward the line $y = 0$ like LSGELU and GELU do. This curving back up toward the line $y = 0$ is a characteristic that LSGELU shares with GELU, where LSGELU approaches the line $y = 0$ when x reaches approximately −4.2, while GELU approaches the line $y = 0$ when x reaches approximately −2.4.

Similarly, to visualize the shape of the RSGELU graph, we have shown it in Fig. 2 with the orange curve. From the figure, we can see that RSGELU curves more to the right than all other graphs if $x > 0$, but it has a shape similar to ReLU if $x < 0$. Fig. 2 shows the shape of RSGELU compared to LSGELU, GELU, ELU, and Swish (In this paper, we do not have any comparative trials with SwiGLU [21]).

The backpropagation process in machine learning requires the derivative of the activation function. Therefore, we derive the derivative of LSGELU, obtaining the red dotted line graph as shown in Fig. 3.

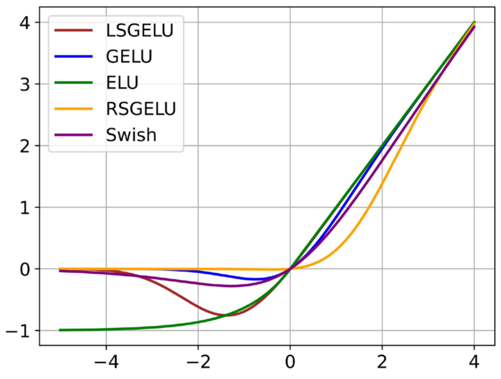


Fig. 2. The LSGELU, GELU, ELU, RSGELU, and Swish.

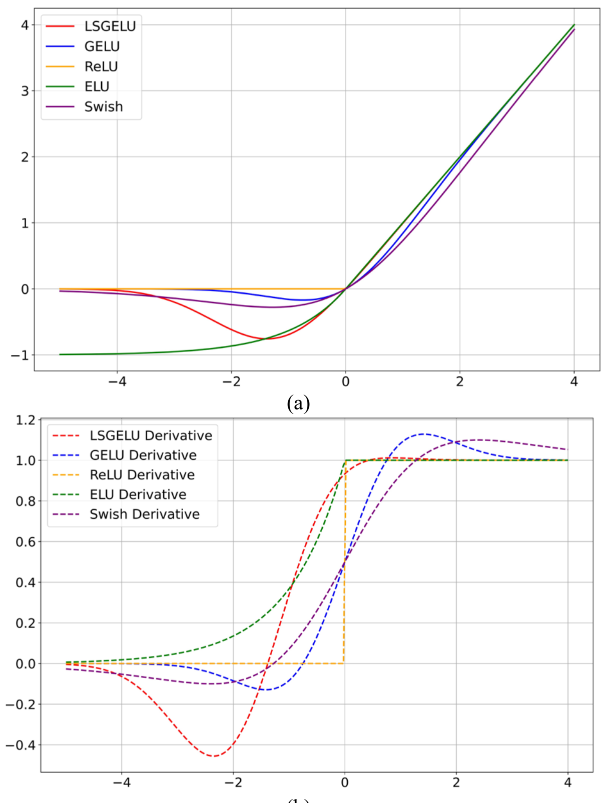
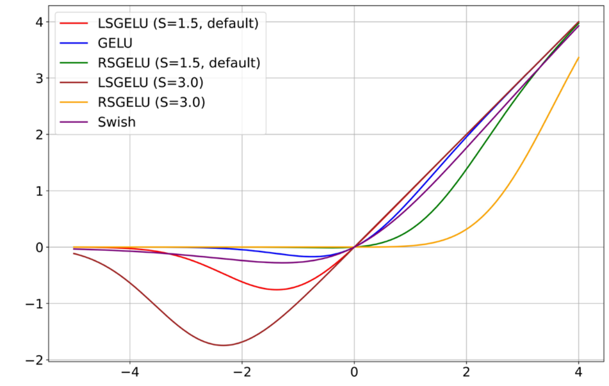


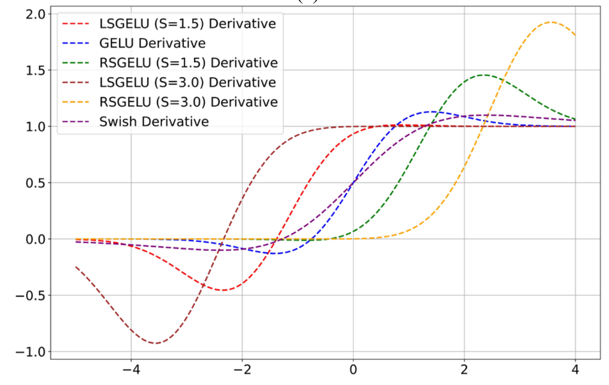
Fig. 3. Comparison of graph shapes of LSGELU, GELU, ReLU, ELU, and Swish with their derivatives. (a) activation functions; (b) derivatives of activation functions.

The graph in Fig. 3 shows the derivative of LSGELU compared with derivatives of other activation functions. When examining the shape of the LSGELU derivative compared to the GELU and Swish derivatives, we find both similarities and differences. The similarities include that both have S-shaped curves that are smoother than the ELU derivative (and smoother than the ReLU derivative) because at the point where $x = 0$, the ELU derivative graph has a sharp corner. The difference between LSGELU derivative and GELU derivative is that the LSGELU derivative approaches a value of 1 when $x = 0$ or $x > 0$.

For LSGELU with $S = 3.0$ as shown in the brown line in Fig. 4, it curves downward the most (if $x < 0$) compared to the other curves, but if $x > 0$ it is closer to the curve of $y = x$ function. For RSGELU with $S = 3.0$ as shown in the orange line (Fig. 4), it curves to the right more than the curve of RSGELU with $S = 1.5$ if $x > 0$, but if $x < 0$ it is closer to the x-axis.



(a)



(b)

Fig. 4. Comparison of the shapes of graphs of LSGELU ($S = 1.5$, default), LSGELU ($S = 3.0$), RSGELU ($S = 1.5$, default), RSGELU ($S = 3.0$), GELU, and Swish, including their derivatives. (a) activation functions; (b) derivatives of activation functions.

Shifting the bell curve to the left increases the area under the left half of the graph. Shifting it 1.5 units to the left gives the left half area of the bell curve 93.32%, as shown in the brown the bell curve on the left of Fig. 5 (if the left half area is 99.95%, it corresponds to $S = 3.290$; if the left half area is 99.00%, it corresponds to $S = 2.326$; if the left half area is 90.00%, it corresponds to $S = 1.281$; if the left half area is 75.00%, it corresponds to $S = 0.674$). From Fig. 4, when we integrate the bell curve function, we get an S-shaped function of the area under the bell curve, shown by the dashed line. And if we multiply the S-shaped graph by x, we get an Activation Function, such as GELU shown by the blue dotted line and LSGELU shown by the brown dotted line.

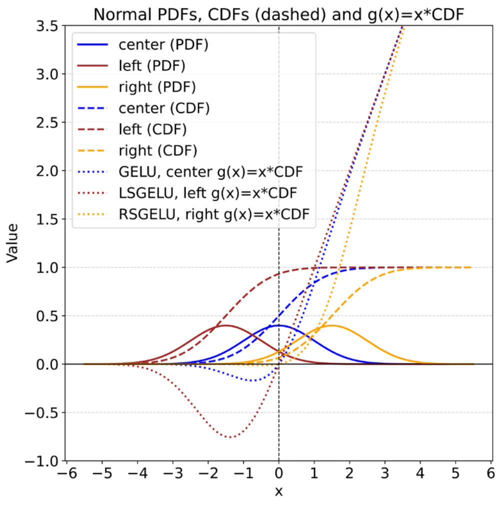


Fig. 5. Bell curves shifted left and right, areas under the curves, and activation functions.

RSGELU is an improvement on LSGELU in that instead of shifting the bell curve to the left, we shift it to the right. After integrating to obtain the area under the curve and multiplying by x, we obtain the RSGELU equation, as shown in Eq. (6) below (its shape is shown in Fig. 4, the orange dotted line).

$$\begin{aligned}\mathrm{RSGELU}(x) &= x \times P\big(X \leq (x - 1.5)\big)\\ &= x \times \Phi(x - 1.5)\\ &= x \times \tfrac{1}{2}\,(1 + \mathrm{erf}((x - 1.5)/\sqrt{2}))\end{aligned} \tag{6}$$

In the next section, we will experiment with LSGELU and RSGELU on different types of data sets.

IV. LSGELU Experiments and Discussion

In this section, we experiment with LSGELU and RSGELU on four different types of datasets, which consist of MNIST, Canadian Institute for Advanced Research-100 classes (CIFAR-100), Street View House Numbers (SVHN), and CIFAR-10.

A. *Experiments with MNIST*

In this experiment, we conducted experiments to measure accuracy values by experimenting with the MNIST (Modified National Institute of Standards and Technology) dataset [11], which is a widely used collection of sample images of digital characters sized 28×28 pixels that is extensively used for testing and comparing the performance of image classification models. This dataset consists of 70,000 images divided into training and testing sets that are popularly used in research on character recognition and image processing.

Firstly, we used only a single hidden layer and employed just 32 nodes (on the hidden layer). The experimental results clearly demonstrate that LSGELU was able to achieve higher accuracy values than GELU, ReLU, and ELU, as shown in Fig. 6.

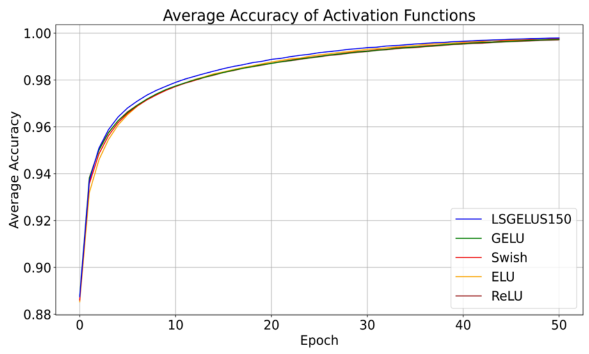


Fig. 6. Experimental results measuring accuracy values with MNIST data (32 nodes on 1 hidden layer).

In addition to providing higher accuracy values than other activation functions, we found that LSGELU also yields lower loss values, as shown in Fig. 7.

We conducted an additional experiment by repeating the experiment 30 times to measure the Accuracy of each Activation Function and then calculated the mean and SD. The experimental results are shown in Table AI in Appendix. We also compared LSGELU with GELU to show that LSGELU gives higher Accuracy than GELU by showing the confidence interval, set to $\alpha = 0.1$ (90%) as shown in Table I.

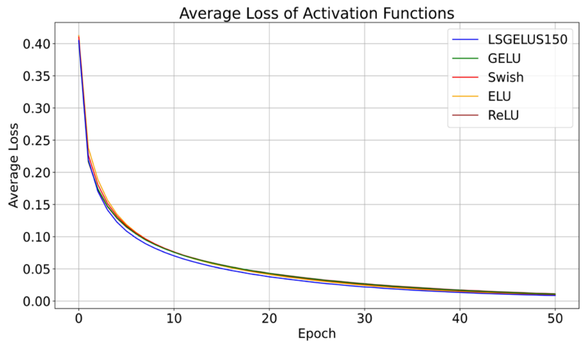


Fig. 7. Experimental results measuring loss values with MNIST data (32 nodes on 1 hidden layer).

The Python source code used for testing LSGELU that produced the experimental results shown in Figs. 6 and 7 is available in the Appendix. Readers can take this source code and modify it to test LSGELU's performance in other environments, such as increasing or decreasing the number of nodes in the hidden layer or even increasing the number of hidden layer levels.

TABLE I. PERFORMANCE COMPARISON OF LSGELU VS GELU WITH 1 HIDDEN LAYER, 32 NODES/HIDDEN LAYER, 51 EPOCHS, $A = 0.1$, N1 = 30, N2 = 30

Ep	Mean1	Mean2	ci_lower	ci_upper	*p*-value	Conclusion
1	0.88746	0.88811	(0.00186)	0.00055	0.282785	no sig
2	0.93709	0.93801	(0.00202)	0.00018	0.098329	no sig
3	0.95097	0.94996	0.00013	0.00189	0.025871	A1 > A2
4	0.95892	0.95743	0.00071	0.00226	0.000326	A1 > A2
5	0.96434	0.96252	0.00110	0.00253	4.74E−06	A1 > A2
6	0.96821	0.96617	0.00138	0.00269	1.02E−07	A1 > A2
7	0.97109	0.96918	0.00125	0.00256	3.47E−07	A1 > A2
8	0.97366	0.97178	0.00123	0.00253	3.47E−07	A1 > A2
9	0.97573	0.97399	0.00111	0.00237	1.22E−06	A1 > A2
10	0.97745	0.97584	0.00101	0.00223	2.52E−06	A1 > A2
11	0.97914	0.97740	0.00117	0.00232	1.56E−07	A1 > A2
12	0.98045	0.97872	0.00118	0.00229	6.38E−08	A1 > A2
13	0.98170	0.98003	0.00113	0.00222	1.14E−07	A1 > A2
14	0.98285	0.98107	0.00125	0.00232	2.00E−08	A1 > A2
15	0.98397	0.98222	0.00121	0.00228	2.61E−08	A1 > A2
16	0.98498	0.98318	0.00127	0.00233	8.25E−09	A1 > A2
17	0.98584	0.98408	0.00121	0.00231	4.69E−08	A1 > A2
18	0.98660	0.98500	0.00109	0.00211	5.01E−08	A1 > A2
19	0.98747	0.98575	0.00121	0.00222	6.40E−09	A1 > A2
20	0.98809	0.98649	0.00113	0.00208	6.76E−09	A1 > A2
21	0.98885	0.98717	0.00120	0.00216	3.20E−09	A1 > A2
22	0.98942	0.98781	0.00109	0.00211	5.66E−08	A1 > A2
23	0.99009	0.98835	0.00131	0.00217	6.53E−11	A1 > A2
24	0.99070	0.98906	0.00115	0.00213	1.10E−08	A1 > A2
25	0.99114	0.98953	0.00115	0.00207	3.65E−09	A1 > A2
26	0.99176	0.99003	0.00125	0.00222	2.34E−09	A1 > A2
27	0.99215	0.99067	0.00104	0.00193	1.15E−08	A1 > A2
28	0.99261	0.99109	0.00108	0.00197	6.47E−09	A1 > A2
29	0.99311	0.99144	0.00125	0.00211	2.15E−10	A1 > A2
30	0.99349	0.99194	0.00111	0.00199	3.00E−09	A1 > A2
31	0.99379	0.99230	0.00106	0.00193	6.86E−09	A1 > A2
32	0.99411	0.99270	0.00101	0.00182	4.92E−09	A1 > A2
33	0.99454	0.99301	0.00112	0.00194	6.68E−10	A1 > A2
34	0.99480	0.99357	0.00085	0.00160	3.01E−08	A1 > A2
35	0.99509	0.99377	0.00091	0.00173	4.54E−08	A1 > A2
36	0.99541	0.99417	0.00089	0.00161	4.95E−09	A1 > A2
37	0.99563	0.99429	0.00092	0.00176	5.38E−08	A1 > A2
38	0.99597	0.99469	0.00092	0.00164	2.45E−09	A1 > A2
39	0.99608	0.99500	0.00070	0.00146	6.51E−07	A1 > A2
40	0.99640	0.99532	0.00074	0.00142	4.99E−08	A1 > A2
41	0.99655	0.99548	0.00074	0.00141	6.33E−08	A1 > A2

42	0.99671	0.99574	0.00061	0.00132	1.26E−06	A1 > A2
43	0.99700	0.99597	0.00074	0.00133	4.46E−09	A1 > A2
44	0.99710	0.99615	0.00063	0.00128	3.63E−07	A1 > A2
45	0.99725	0.99631	0.00062	0.00126	2.89E−07	A1 > A2
46	0.99742	0.99652	0.00058	0.00122	5.31E−07	A1 > A2
47	0.99748	0.99669	0.00050	0.00107	8.31E−07	A1 > A2
48	0.99762	0.99687	0.00047	0.00104	1.57E−06	A1 > A2
49	0.99776	0.99706	0.00041	0.00099	1.10E−05	A1 > A2
50	0.99790	0.99714	0.00050	0.00101	1.72E−07	A1 > A2
51	0.99791	0.99725	0.00038	0.00095	2.48E−05	A1 > A2

Additional information about Table I is in Table A.II in Appendix; A1 = LSGELU, A2 = GELU, "no sig" = no significant difference.

We performed the ranking using the mean accuracy values for epochs 46 to 51 at $\alpha = 0.05$. The results were as follows:

Overall ranking by mean accuracy (1 is best):
(1) LSGELUS150 (mean_accuracy = 0.997635);
(2) ELU (mean_accuracy = 0.997365);
(3) Swish (mean_accuracy = 0.997096);
(4) GELU (mean_accuracy = 0.996856);
(5) ReLU (mean_accuracy = 0.996499).

B. Experiments with CIFAR-100 and CNN

In this subsection, we experiment with CIFAR-100 data using Convolutional Neural Network (CNN) to increase the model complexity. We use 256 nodes on each of the 5 hidden layers. CIFAR-100 is a colour image dataset of objects such as animals, objects, and vehicles divided into 100 classes. We conducted 12 experiments. The experimental results are shown in Fig. 8. The black line is RSGELU with S = 1.5 (in the figure, it is denoted RSGELUS150). While at low epochs, the accuracy is lower than all Activation Functions, but when it progresses to around 70 epochs, it shows better performance than other Activation Functions. We averaged the data from this set of experiments and plotted it graphs so that each Activation Function has only one graph, as shown in Fig. 9.

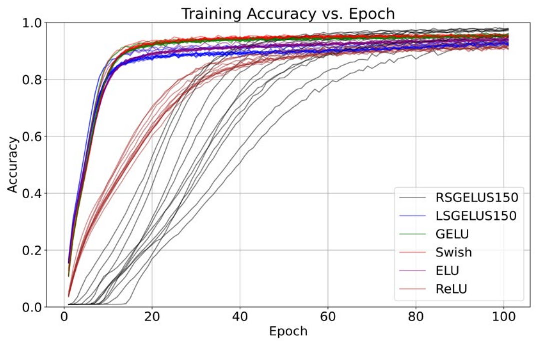


Fig. 8. Experimental results measuring accuracy values with CIFAR-100 dataset using CNN (256 nodes on each of the 5 hidden layer).

From Fig. 9 in the later epochs, the mean of RSGELU, GELU and Swish are very close to each other, so we use statistics to test. We take the data from epoch 99, 100 and 101 and compete them one pair at a time. If any pair is not statistically different (using t-test, $\alpha = 0.05$), the result is "no_consistent_difference". If there is a win or lose, it is shown in the table. The results are shown in Table II, where the matching results are displayed in pairs to determine the top score.

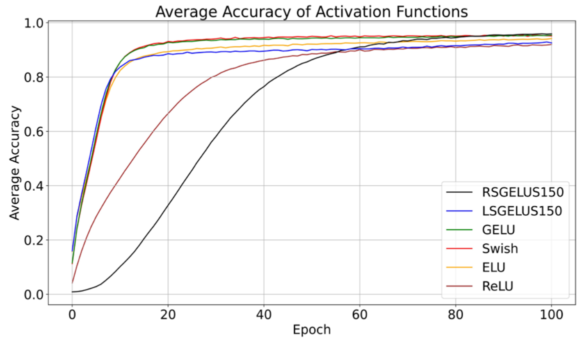


Fig. 9. Experimental results measuring accuracy values with CIFAR-100 dataset using CNN (256 nodes on each of the 5 hidden layer).

TABLE II. EXPERIMENTAL RESULTS MEASURING ACCURACY VALUES WITH THE CIFAR-100 DATASET USING CNN

Pair	Result
RSGELUS150 vs LSGELUS150	RSGELUS150 > LSGELUS150
RSGELUS150 vs GELU	no_consistent_difference
RSGELUS150 vs Swish	no_consistent_difference
RSGELUS150 vs ELU	RSGELUS150 > ELU
RSGELUS150 vs ReLU	RSGELUS150 > ReLU
LSGELUS150 vs GELU	GELU > LSGELUS150
LSGELUS150 vs Swish	Swish > LSGELUS150
LSGELUS150 vs ELU	ELU > LSGELUS150
LSGELUS150 vs ReLU	LSGELUS150 > ReLU
GELU vs Swish	GELU > Swish
GELU vs ELU	GELU > ELU
GELU vs ReLU	GELU > ReLU
Swish vs ELU	Swish > ELU
Swish vs ReLU	Swish > ReLU
ELU vs ReLU	ELU > ReLU

Overall ranking by mean accuracy (1 is best):
(1) RSGELUS150 (mean_accuracy = 0.958550);
(2) GELU (mean_accuracy = 0.953628);
(3) Swish (mean_accuracy = 0.951636);
(4) ELU (mean_accuracy = 0.940430);
(5) LSGELU150 (mean_accuracy = 0.927334);
(6) ReLU (mean_accuracy = 0.918854).

From the experimental results in the table, it can be seen that LSGELU (with S = 1.5) does not give results that are as good as other Activation Functions except ReLU which has the lowest performance. RSGELU gives accuracy at the end of the epoch, comparable to GELU and Swish. If we consider the mean accuracy, RSGELU is the Activation Function that gives the highest accuracy value.

C. Experiments with SVHN

In this subsection, we conducted experiments on the SVHN dataset, which is a colour image of house numbers divided into 10 classes.

The experimental results are shown in Fig. 10.

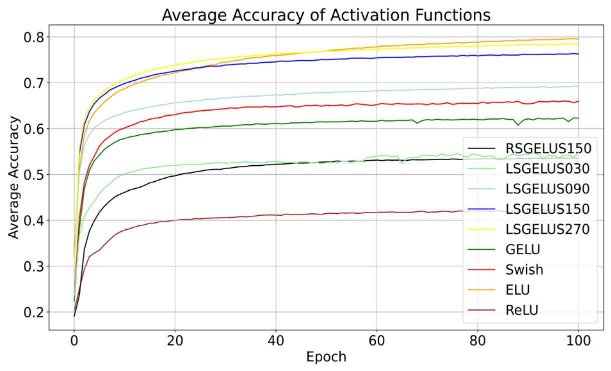


Fig. 10. Experimental results measuring accuracy values with SVHN dataset (32 nodes on each of the 6 hidden layers).

Varying the distance of shifting the left (or right) side of the bell curve has a similar effect to varying hyperparameters such as learning rate and batch size. We have conducted experiments and presented additional experimental results to show that shifting the left side by different distances, such as 0.3, 0.9, 1.5, and 2.7, gives different results.

We also conducted experiments with LSGELU with S values of 0.3, 0.9, 1.5, and 2.7. The results showed that, although LSGELU with S value of 2.7 was not the highest ranking, it performed well in the second place, closely matching ELU's ranking.

Considering Fig. 4, the brown line represents LSGELU with S value of 3.0. The shape is similar to that of LSGELU with S value of 2.7 (not shown in this paper). The graph has a thin shape on the left side of the y-axis ($x < 0$), similar to the ELU graph in Fig. 2 (green line), such as a very steep downward curve.

Consider Fig. 10 again. In addition to the good results of LSGELU shifted significantly to the left, such as 2.7, we also found that LSGELU (S = 1.5, default) performed better than GELU, which did not shift to the left at all. Conversely, RSGELU, with the black line, performed poorly. However, in this experiment, RSGELU was not the worst performer, as the worst performer was ReLU.

D. *Experiments with CIFAR-10*

In this subsection, we conduct experiments on the CIFAR-10 dataset. CIFAR-10 is a color image dataset, containing images of animals, objects, and vehicles, just like CIFAR-100. Instead of dividing the data into 100 classes, it uses only 10. We use 32 nodes on each of the 6 hidden layers, as in the previous subsection. The experimental results are similar to those of SVHN, with each activation function exhibiting distinctly different results, as shown in Fig. 11.

From Fig. 11 it can be seen that RSGELU, ReLU and GELU do not give good results as other Activation Functions, while the Activation Functions that give good accuracy are LSGELU with large left shifts, such as LSGELU with S = 2.7 and ELU.

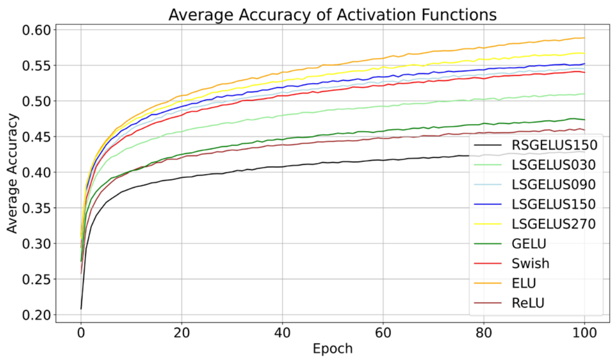


Fig. 11. Experimental results measuring accuracy values with CIFAR-10 dataset (32 nodes on each of the 6 hidden layers).

E. *Theoretical Analysis*

Based on the experiments reported in the preceding subsections, we find that LSGELU is better suited for simple neural networks—few layers, few units per layer, no CNN components—and datasets with a small number of classes. In contrast, RSGELU is more suitable for complex neural networks with many hidden layers and many units per layer, and for datasets with many classes (e.g., CIFAR-100); the gains are even more pronounced with CNNs. Shifting the bell curve to the left in LSGELU produces a deeper "shallow-negative" basin over approximately −3 to 0 than GELU or Swish. Conversely, shifting the bell curve to the right in RSGELU yields a deeper "shallow-positive" basin over approximately 0 to 3, also deeper than that of GELU or Swish.

Attenuating the negative side between 0 and −3 suits small models because it avoids hard zeroing as in ReLU. Mildly preserving small negative signals prevents neuron death, maintains a trickle of gradient, and provides sufficient flexibility for simple tasks (e.g., MNIST) without requiring deep capacity. Training becomes steadier while still emphasizing the principal patterns.

Attenuating the positive side between 0 and 3 benefits large, deep, and convolutional networks by easing the throttle on small positive activations that would otherwise amplify across many layers. This stabilizes training, keeps activation statistics balanced, and prioritizes only truly salient features. In practice, this tends to improve generalization on complex, many-class datasets such as CIFAR-100.

However, over-attenuation on either side harms performance. Useful cues get pushed too low, gradients weaken and fade across layers, optimization stalls or plateaus, and normalization/optimizers operate suboptimally. The model's discriminative power drops, learning slows, and test results degrade—even compared with standard activations. The sweet spot is moderate attenuation on each side, not too much.

V. Conclusion and Further Works

In this research, we have presented a new activation function called LSGELU. Furthermore, we also present RSGELU. We conducted experiments with MNIST, SVHN, CIFAR10, and CIFAR100 (with Convolution Neural Network) data and found that LSGELU performs better than many existing traditional activation functions such as GELU, ReLU, and ELU in some specific environments (e.g., with a small number of hidden layers and a small number of nodes in those layers). This superior performance means that during model training, it helps the model learn more effectively, with LSGELU's accuracy being able to reach 1.00 fastest while having the lowest loss compared to other activation functions. From the experimental results, we also discovered that if we use too many hidden layers or if we have too many nodes per hidden layer, this can result in a tendency for LSGELU performance to become lower than GELU. Therefore, we can see that LSGELU has advantages that can be applied in practical applications, such as use with models that have moderate complexity, such as small devices with limited memory capacity. In addition to experimenting with LSGELU, we also conducted experiments with RSGELU. The experimental results showed that for simple models, with a small number of hidden layers and a small number of nodes in those layers, without using CNNs, RSGELU performs poorly. However, if experimenting with more complex models and using CNNs, RSGELU performs surprisingly well. In future research, we plan to pilot LSGELU and RSGELU using the chest X-ray images (Pneumonia) dataset.

Appendix: Source Code and Experimental Results for LSGELU Performance Comparison

A simple source code example for running on Jupiter Notebook to demonstrate experimental results in a rough and rapid manner, requiring approximately 5 minutes of processing time on an Intel Core i5 3.10 GHz CPU, which produces the experimental results shown in Figs. 6 and 7, has been presented in Listing A1.

```
#--- 2025-10-30 15-41 – by Dr. Thawatchai Chomsiri
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
import datetime

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train = (x_train.astype(‘float32’)/255.) / 1.00
x_test = (x_test.astype(‘float32’)/255.) / 1.00
x_train = x_train.reshape(-1, 784)
x_test = x_test.reshape(-1, 784)

def lsgelu(x):
   S = 1.5
   return x * 0.5 * (1 + tf.math.erf((x + S) / tf.sqrt(2.0)))

activations_list = {
   “LSGELU”: lsgelu,
   “GELU”: tf.nn.gelu,
   “Swish”: tf.nn.swish,
   “ELU”: tf.keras.activations.elu,
   “ReLU”: tf.keras.activations.relu,
}

print(f”\nBEGIN at {datetime.datetime.now().strftime(‘%Y-%m-%d %H:%M:%S’)}”)
results = {}
for act_name, act_fn in activations_list.items():
   print(f”Running: Activation={act_name}”)
   model = tf.keras.Sequential([
      tf.keras.layers.Dense(32, activation=act_fn, input_shape=(784,)),
      tf.keras.layers.Dense(10, activation=‘softmax’)
   ])
   model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss=‘sparse_categorical_crossentropy’,
              metrics=[‘accuracy’])
   history = model.fit(x_train, y_train, epochs=51, batch_size=64, verbose=1)
   key = f”{act_name}”
   results[key] = history

print(f”\nEND at {datetime.datetime.now().strftime(‘%Y-%m-%d %H:%M:%S’)}”)
plt.figure(figsize=(15,8))
for key, history in results.items():
   plt.plot(history.history[‘accuracy’], label=key)
plt.xlabel(‘Epochs’)
plt.ylabel(‘Accuracy’)
plt.title(‘Activation Function Performance on MNIST (Accuracy)’)
plt.legend()
plt.grid(True)
plt.show()
plt.figure(figsize=(15,8))
for key, history in results.items():
    plt.plot(history.history[‘loss’], label=key)
plt.xlabel(‘Epochs’)
plt.ylabel(‘Loss’)
plt.title(‘Activation Function Performance on MNIST (Loss)’)
plt.legend()
plt.grid(True)
plt.show()
```

Listing A1. Source code for simple LSGELU testing with graph plotting for performance comparison.

TABLE AI. EXPERIMENTAL RESULTS WITH MNIST WITH A MODEL SIZE OF 1 HIDDEN LAYER, 32 NODES PER HIDDEN LAYER

Epoch	Activation Function									
	LSGELU		GELU		Swish		ELU		ReLU	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
1	0.88746	0.00237	0.88811	0.00230	0.88603	0.00192	0.88565	0.00240	0.88591	0.00334
2	0.93709	0.00223	0.93801	0.00201	0.93580	0.00204	0.93208	0.00209	0.93801	0.00145
3	0.95097	0.00164	0.94996	0.00177	0.94866	0.00185	0.94587	0.00201	0.95005	0.00138
4	0.95892	0.00136	0.95743	0.00163	0.95647	0.00181	0.95451	0.00166	0.95738	0.00118
5	0.96434	0.00119	0.96252	0.00155	0.96198	0.00176	0.96055	0.00143	0.96237	0.00100
6	0.96821	0.00099	0.96617	0.00150	0.96598	0.00156	0.96496	0.00125	0.96610	0.00117
7	0.97109	0.00100	0.96918	0.00147	0.96910	0.00142	0.96850	0.00102	0.96899	0.00113
8	0.97366	0.00104	0.97178	0.00143	0.97168	0.00139	0.97129	0.00112	0.97147	0.00104
9	0.97573	0.00096	0.97399	0.00144	0.97389	0.00139	0.97352	0.00098	0.97361	0.00090
10	0.97745	0.00092	0.97584	0.00139	0.97579	0.00130	0.97552	0.00103	0.97548	0.00084
11	0.97914	0.00087	0.97740	0.00131	0.97738	0.00129	0.97719	0.00100	0.97688	0.00088
12	0.98045	0.00086	0.97872	0.00124	0.97884	0.00118	0.97867	0.00092	0.97846	0.00092
13	0.98170	0.00086	0.98003	0.00122	0.97999	0.00131	0.98015	0.00084	0.97968	0.00084
14	0.98285	0.00080	0.98107	0.00123	0.98119	0.00113	0.98126	0.00089	0.98072	0.00095
15	0.98397	0.00084	0.98222	0.00120	0.98230	0.00112	0.98248	0.00095	0.98177	0.00090
16	0.98498	0.00089	0.98318	0.00114	0.98334	0.00124	0.98330	0.00092	0.98287	0.00090
17	0.98584	0.00085	0.98408	0.00124	0.98419	0.00117	0.98431	0.00085	0.98364	0.00087
18	0.98660	0.00085	0.98500	0.00110	0.98511	0.00109	0.98522	0.00073	0.98460	0.00098
19	0.98747	0.00096	0.98575	0.00100	0.98588	0.00105	0.98596	0.00074	0.98532	0.00091
20	0.98809	0.00086	0.98649	0.00097	0.98662	0.00107	0.98677	0.00078	0.98603	0.00089
21	0.98885	0.00081	0.98717	0.00103	0.98729	0.00106	0.98750	0.00080	0.98679	0.00083
22	0.98942	0.00083	0.98781	0.00112	0.98797	0.00106	0.98814	0.00083	0.98740	0.00092
23	0.99009	0.00074	0.98835	0.00092	0.98862	0.00109	0.98891	0.00073	0.98794	0.00083
24	0.99070	0.00082	0.98906	0.00106	0.98915	0.00108	0.98923	0.00080	0.98858	0.00077
25	0.99114	0.00079	0.98953	0.00098	0.98972	0.00098	0.99004	0.00075	0.98912	0.00090
26	0.99176	0.00080	0.99003	0.00106	0.99028	0.00099	0.99053	0.00080	0.98958	0.00088
27	0.99215	0.00073	0.99067	0.00096	0.99079	0.00095	0.99100	0.00073	0.99019	0.00081
28	0.99261	0.00072	0.99109	0.00098	0.99123	0.00087	0.99157	0.00080	0.99053	0.00080
29	0.99311	0.00068	0.99144	0.00095	0.99176	0.00093	0.99193	0.00068	0.99110	0.00089
30	0.99349	0.00068	0.99194	0.00097	0.99214	0.00088	0.99238	0.00072	0.99150	0.00078
31	0.99379	0.00069	0.99230	0.00097	0.99253	0.00095	0.99291	0.00064	0.99190	0.00079
32	0.99411	0.00066	0.99270	0.00090	0.99298	0.00086	0.99321	0.00065	0.99229	0.00087
33	0.99454	0.00068	0.99301	0.00089	0.99328	0.00090	0.99357	0.00070	0.99268	0.00072
34	0.99480	0.00056	0.99357	0.00086	0.99371	0.00086	0.99401	0.00060	0.99303	0.00080
35	0.99509	0.00063	0.99377	0.00093	0.99400	0.00089	0.99428	0.00067	0.99340	0.00075
36	0.99541	0.00060	0.99417	0.00078	0.99424	0.00075	0.99456	0.00068	0.99366	0.00067
37	0.99563	0.00059	0.99429	0.00098	0.99466	0.00076	0.99498	0.00066	0.99396	0.00074
38	0.99597	0.00067	0.99469	0.00073	0.99494	0.00079	0.99521	0.00065	0.99419	0.00070
39	0.99608	0.00058	0.99500	0.00087	0.99515	0.00073	0.99551	0.00061	0.99460	0.00066
40	0.99640	0.00055	0.99532	0.00075	0.99541	0.00072	0.99566	0.00053	0.99485	0.00072
41	0.99655	0.00044	0.99548	0.00080	0.99574	0.00070	0.99595	0.00062	0.99511	0.00067
42	0.99671	0.00059	0.99574	0.00077	0.99597	0.00071	0.99637	0.00051	0.99534	0.00074
43	0.99700	0.00046	0.99597	0.00066	0.99620	0.00066	0.99644	0.00047	0.99548	0.00061
44	0.99710	0.00045	0.99615	0.00076	0.99630	0.00061	0.99667	0.00061	0.99570	0.00062
45	0.99725	0.00050	0.99631	0.00072	0.99662	0.00062	0.99689	0.00049	0.99605	0.00055
46	0.99742	0.00055	0.99652	0.00068	0.99677	0.00061	0.99695	0.00048	0.99612	0.00065
47	0.99748	0.00046	0.99669	0.00062	0.99689	0.00055	0.99715	0.00050	0.99633	0.00050
48	0.99762	0.00053	0.99687	0.00056	0.99716	0.00067	0.99742	0.00044	0.99650	0.00054
49	0.99776	0.00050	0.99706	0.00061	0.99724	0.00061	0.99763	0.00044	0.99674	0.00057
50	0.99790	0.00044	0.99714	0.00054	0.99742	0.00053	0.99767	0.00041	0.99680	0.00049
51	0.99791	0.00037	0.99725	0.00068	0.99751	0.00052	0.99778	0.00047	0.99698	0.00050

TABLE AII. More Information about the Performance Comparison of LSGELU vs GELU with 1 Hidden Layer, 32 Nodes/Hidden Layer, 51 Epochs, *A* = 0.1, n1 = 30, n2 = 30

Epoch	n1	n2	Mean1	Mean2	Mean_diff	ci_lower	ci_upper	*p*-value	Conclusion
1	30	30	0.88746	0.88811	(0.00065)	(0.00186)	0.00055	0.2827851	no_significant_difference
2	30	30	0.93709	0.93801	(0.00092)	(0.00202)	0.00018	0.0983294	no_significant_difference
3	30	30	0.95097	0.94996	0.00101	0.00013	0.00189	0.0258706	LSGELUS150 > GELU
4	30	30	0.95892	0.95743	0.00149	0.00071	0.00226	0.0003257	LSGELUS150 > GELU
5	30	30	0.96434	0.96252	0.00181	0.00110	0.00253	4.74E−06	LSGELUS150 > GELU
6	30	30	0.96821	0.96618	0.00204	0.00138	0.00269	1.02E−07	LSGELUS150 > GELU
7	30	30	0.97109	0.96918	0.00190	0.00125	0.00256	3.47E−07	LSGELUS150 > GELU
8	30	30	0.97366	0.97178	0.00188	0.00123	0.00253	3.47E−07	LSGELUS150 > GELU
9	30	30	0.97573	0.97399	0.00174	0.00111	0.00237	1.22E−06	LSGELUS150 > GELU
10	30	30	0.97745	0.97584	0.00162	0.00101	0.00223	2.52E−06	LSGELUS150 > GELU
11	30	30	0.97914	0.97740	0.00175	0.00117	0.00232	1.56E−07	LSGELUS150 > GELU
12	30	30	0.98045	0.97872	0.00173	0.00118	0.00229	6.38E−08	LSGELUS150 > GELU
13	30	30	0.98170	0.98003	0.00167	0.00113	0.00222	1.14E−07	LSGELUS150 > GELU
14	30	30	0.98285	0.98107	0.00179	0.00125	0.00232	2.00E−08	LSGELUS150 > GELU
15	30	30	0.98397	0.98222	0.00175	0.00121	0.00228	2.61E−08	LSGELUS150 > GELU
16	30	30	0.98498	0.98318	0.00180	0.00127	0.00233	8.25E−09	LSGELUS150 > GELU
17	30	30	0.98584	0.98408	0.00176	0.00121	0.00231	4.69E−08	LSGELUS150 > GELU
18	30	30	0.98660	0.98500	0.00160	0.00109	0.00211	5.01E−08	LSGELUS150 > GELU
19	30	30	0.98747	0.98575	0.00172	0.00121	0.00222	6.40E−09	LSGELUS150 > GELU
20	30	30	0.98809	0.98649	0.00160	0.00113	0.00208	6.76E−09	LSGELUS150 > GELU
21	30	30	0.98885	0.98717	0.00168	0.00120	0.00216	3.20E−09	LSGELUS150 > GELU
22	30	30	0.98942	0.98781	0.00160	0.00109	0.00211	5.66E−08	LSGELUS150 > GELU
23	30	30	0.99009	0.98835	0.00174	0.00131	0.00217	6.53E−11	LSGELUS150 > GELU
24	30	30	0.99070	0.98906	0.00164	0.00115	0.00213	1.10E−08	LSGELUS150 > GELU
25	30	30	0.99114	0.98953	0.00161	0.00115	0.00207	3.65E−09	LSGELUS150 > GELU
26	30	30	0.99176	0.99003	0.00173	0.00125	0.00222	2.34E−09	LSGELUS150 > GELU
27	30	30	0.99215	0.99067	0.00148	0.00104	0.00193	1.15E−08	LSGELUS150 > GELU
28	30	30	0.99261	0.99109	0.00152	0.00108	0.00197	6.47E−09	LSGELUS150 > GELU
29	30	30	0.99311	0.99144	0.00168	0.00125	0.00211	2.15E−10	LSGELUS150 > GELU
30	30	30	0.99349	0.99194	0.00155	0.00111	0.00199	3.00E−09	LSGELUS150 > GELU
31	30	30	0.99379	0.99230	0.00150	0.00106	0.00193	6.86E−09	LSGELUS150 > GELU
32	30	30	0.99411	0.99270	0.00142	0.00101	0.00182	4.92E−09	LSGELUS150 > GELU
33	30	30	0.99454	0.99301	0.00153	0.00112	0.00194	6.68E−10	LSGELUS150 > GELU
34	30	30	0.99480	0.99357	0.00123	0.00085	0.00160	3.01E−08	LSGELUS150 > GELU
35	30	30	0.99509	0.99377	0.00132	0.00091	0.00173	4.54E−08	LSGELUS150 > GELU
36	30	30	0.99541	0.99417	0.00125	0.00089	0.00161	4.95E−09	LSGELUS150 > GELU
37	30	30	0.99563	0.99429	0.00134	0.00092	0.00176	5.38E−08	LSGELUS150 > GELU
38	30	30	0.99597	0.99469	0.00128	0.00092	0.00164	2.45E−09	LSGELUS150 > GELU
39	30	30	0.99608	0.99500	0.00108	0.00070	0.00146	6.51E−07	LSGELUS150 > GELU
40	30	30	0.99640	0.99532	0.00108	0.00074	0.00142	4.99E−08	LSGELUS150 > GELU
41	30	30	0.99655	0.99548	0.00108	0.00074	0.00141	6.33E−08	LSGELUS150 > GELU
42	30	30	0.99671	0.99574	0.00097	0.00061	0.00132	1.26E−06	LSGELUS150 > GELU
43	30	30	0.99700	0.99597	0.00104	0.00074	0.00133	4.46E−09	LSGELUS150 > GELU
44	30	30	0.99710	0.99615	0.00095	0.00063	0.00128	3.63E−07	LSGELUS150 > GELU
45	30	30	0.99725	0.99631	0.00094	0.00062	0.00126	2.89E−07	LSGELUS150 > GELU
46	30	30	0.99742	0.99652	0.00090	0.00058	0.00122	5.31E−07	LSGELUS150 > GELU
47	30	30	0.99748	0.99669	0.00078	0.00050	0.00107	8.31E−07	LSGELUS150 > GELU
48	30	30	0.99762	0.99687	0.00075	0.00047	0.00104	1.57E−06	LSGELUS150 > GELU
49	30	30	0.99776	0.99706	0.00070	0.00041	0.00099	1.10E−05	LSGELUS150 > GELU
50	30	30	0.99790	0.99714	0.00076	0.00050	0.00101	1.72E−07	LSGELUS150 > GELU
51	30	30	0.99791	0.99725	0.00066	0.00038	0.00095	2.48E−05	LSGELUS150 > GELU

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

The initial concept for developing the LSGELU and RSGELU formulas was proposed by TC, who also implemented the preliminary testing code before transferring the work to SP for comprehensive experimentation. The experimental results were analyzed by TC. The manuscript was collaboratively written by TC and SP. All authors had approved the final version.

Funding

This research was funded by the Revenue Budget Research Grant, Faculty of Informatics, Mahasarakham University, fiscal year 2025, under the Publication Promotion Program for National and International Journals, Grant No. FT4-04/2568.

Acknowledgment

The authors wish to thank Prof. Chakchai So-In, Ph.D. (Khon Kaen University, Thailand) and Prof. Siba Udgata, Ph.D. (University of Hyderabad, India) for their guidance in this research.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[3] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. the 14th International Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2011, pp. 315–323.

[4] J. He, L. Li, J. Xu *et al.,* "ReLU deep neural networks and linear finite elements," arXiv preprint, arXiv:1807.03973, 2018.

[5] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by Exponential Linear Units (ELUs)," arXiv preprint, arXiv:1511.07289, 2015.

[6] T. Devi and N. Deepa, "A novel intervention method for aspect-based emotion Using Exponential Linear Unit (ELU) activation function in a deep neural network," in *Proc. 2021 5th International Conf. on Intelligent Computing and Control Systems (ICICCS)*, 2021, pp. 1671–1675.

[7] D. Kim, J. Kim, and J. Kim, "Elastic exponential linear units for convolutional neural networks," *Neurocomputing*, vol. 406, pp. 253–266, 2020.

[8] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," arXiv preprint, arXiv:1606.08415, 2016.

[9] D. Hendrycks and K. Gimpel. "A baseline for detecting misclassified and out-of-distribution examples in neural networks," arXiv preprint, arXiv:1610.02136, 2016.

[10] Y. LeCun, L. Bottou, Y. Bengio *et al.,* "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] K. Faber, D. Zurek, M. Pietron *et al.*, "From MNIST to ImageNet and back: Benchmarking continual curriculum learning," *Mach. Learn.,* vol. 113, no. 10, pp. 8137–8164, 2024.

[12] A. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. 30th Int. Conf. Machine Learning (ICML)*, 2013, pp. 1–9.

[13] S. Eger, P. Youssef, and I. Gurevych, "Is it time to swish? Comparing deep learning activation functions across NLP tasks," arXiv preprint, arXiv:1901.02671, 2019.

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

[15] D. Liu, P. Wang, and B. Li, "Mish: A self regularized non-monotonic activation function," arXiv preprint, arXiv:1908.08681, 2019.

[16] H. Zheng, Z. Yang, W. Liu *et al.,* "Improving deep neural networks using softplus units," in *Proc. 2015 International Joint Conf. on Neural Networks (IJCNN)*, 2015, pp. 1–4.

[17] J. Bergstra, G. Desjardins, P. Lamblin *et al.,* "Quadratic polynomials learn better image features," Technical Report, 1337, 2009.

[18] S. Narayan, "The generalized sigmoid activation function: Competitive supervised learning," *Information Sciences,* vol. 99, no. 1–2, pp. 69–82, 1997.

[19] B. Chen, W. Deng, and J. Du, "Noisy SoftMax: Improving the generalization ability of DCNN via postponing the early SoftMax saturation," in *Proc. the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017, pp. 5372–5381.

[20] Desmos Graphing Calculator. *Desmos.* [Online]. Available: https://www.desmos.com/calculator/r0e2biz5dx

[21] F. A. Zidi, D. E. Boukhari, A. Z. Sellam *et al.,* "LoLA-SpecViT: Local attention SwiGLU vision transformer with LoRA for hyperspectral imaging," arXiv preprint, arXiv:2506.17759, 2025.