Neural Computing and Applications (2023) 35:22705–22724
https://doi.org/10.1007/s00521-023-08932-z

ORIGINAL ARTICLE

Deep learning with ExtendeD Exponential Linear Unit (DELU)

Burak Çatalbaş[1] · Ömer Morgül[1]

Received: 29 December 2022 / Accepted: 24 July 2023 / Published online: 16 August 2023

Abstract
Activation functions are crucial parts of artificial neural networks. From the first perceptron created artificially up to today, many functions are proposed. Some of them are currently in common use, such as Rectified Linear Unit (ReLU) and Exponential Linear Unit (ELU) and other ReLU variants. In this article we propose a novel activation function, called ExtendeD Exponential Linear Unit (DELU). After its introduction and presenting its basic properties, by making various simulations with different datasets and architectures, we show that it may perform better than other activation functions in certain cases. While also inheriting most of the good properties of ReLU and ELU, DELU offers an increase of success in comparison with them by slowing the alignment of neurons in early stages of training process. In experiments, DELU performed better than other activation functions in general, for Fashion MNIST, CIFAR-10 and CIFAR-100 classification tasks with different sized Residual Neural Networks (ResNet). Specifically, DELU managed to reduce the error rate by sufficiently high confidence levels in CIFAR datasets in comparison with ReLU and ELU networks. In addition, DELU is compared in an image segmentation example as well. Also, compatibility of DELU is tested with different initializations, and statistical methods are employed to verify these success rates by using Z-score analysis, which may be considered as a different view of success assessment in neural networks.

Keywords Artificial neural networks · Activation functions · Classification · Image segmentation

1 Introduction

Artificial neural networks are the most popular field of machine learning today. With the employment of GPU cards for training and inference, and the rise of big data, the performance of neural networks increased massively. However, there is still some room for improvement due to the nature of neural network structures: Different initialization techniques and different activation functions—as proposed in this paper—can be designed. Activation functions are important parts of the artificial neural networks. From Pitts-McCullogh neuron of 1943 until today, many different activation functions are proposed, and some of them are regarded as 'improvement' due to their significant success in comparison with the others [1]. For the classification and related tasks, namely sigmoid, hyperbolic tangent and then ReLU family members are proposed as an important step toward to the modern neural network architecture [1]. They are mainly used for feed-forward, convolutional [2], recurrent [3], etc. types of neural networks. The former activation functions mentioned above are well known, and the information about them could be found in classical textbooks, e.g., [4]. These activation functions are known to have what is called as 'vanishing gradient' problem which slows down the training process, see e.g., [5].

Mainly to eliminate the vanishing gradient problem, in [6] the authors have started with different alternatives such as Noisy ReLU and their work has evolved to standardized ReLU. It was also claimed that building networks is like an art [7]. This points out to difficulties of the design and, sometimes, lack of clear theoretical explanation for the statistical success of methods employed in neural networks. There are different neuron types, network types and datasets to experiment on, which makes the selection of an appropriate structure even more complex. Convolutional

✉ Burak Çatalbaş
catalbas@ee.bilkent.edu.tr

Ömer Morgül
morgul@ee.bilkent.edu.tr

[1] Department of Electrical and Electronics Engineering, Bilkent University, 06800 Ankara, Turkey

neural networks, for example, need more generalization as seen in [8]. In addition to data augmentation, regularization and network pruning are among the techniques that are employed in convolutional neural networks for better performance [8, 9].

Rectified Linear Unit (ReLU) is first proposed in [6] as a suitable neural network activation function, boosted the training speed and success rates of the multilayer neural networks considerably in various cases. Mostly eliminating the vanishing gradient problem and speeding up the learning algorithms because of the simplicity of its mathematical structure, ReLU is still used in many novel network designs for classification, clustering, recognition tasks by networks such as Inception, GoogLeNet, etc. Nonetheless, its zero part (i.e., the activation becomes zero for negative inputs) named as 'dead zone' [10] causes slowing down for learning process of neural network, in comparison with other activation functions. This problem is referred to as 'dying ReLU problem' in the literature, see, e.g., [5, 11]. Afterward, the alternatives to ReLU have emerged, and an important one is the Exponential Linear Unit (ELU), which is mainly proposed to eliminate the 'dying ReLU problem' mentioned above.

ELU is introduced in [12], and it contains a smoothness addition as seen in Fig. 1. This causes slightly more complexity in exchange of a better performance. Based on good aspects of ReLU variants, ELU offers significant improvement of success rates, especially on certain applications. As an example, results for pattern classification problems with ELU-based networks on various datasets are given in [12]. A faster equivalent is implemented as FELU, see [13]. Also, an interesting and modified version of the latter, namely P+FELU, is proposed in [14].

There are different classifications of activation functions, for example, specialties-based fixed shape vs. trainable shape classification [11], application-based activation function classifications [5], bounded vs. unbounded classification, etc. To mention some of those activation functions, we can label Leaky ReLU and Scaled Exponential Linear Unit (SELU) activation functions. Leaky ReLU introduces a linear part at negative side of the ReLU activation function, with the 'leak' is determined by a parameter just like ELU [15]. This function aims on 'dead zone' problem as well; however, it causes exploding gradient problem for both positive and negative inputs. Softplus activation function is also offered as smooth approximation to ReLU [5]. Another activation function is SELU, which modifies the negative input part of ReLU for self-normalization procedure to learn better. However, this activation function is aimed to be used with a specific weight initialization method [15]. Recently introduced "Swish Activation Function" is a modification of classical sigmoidal function [16]. Remarkable differences for this activation function are suggested as its non-monotonic nature and probable slow learning encountered as the input of the activation function decreases, due to its derivative decreasing to zero as input goes to negative infinity [16]. Later, various modifications of Swish and its variants are proposed such as Shape Autotuning Activation Function [17]. In contrast, two different and novel activation functions are proposed in a comparative paper, namely IpLU and AbsLU, which are smooth and piecewise activation functions, respectively [18]. There are novel two-part piecewise activation functions as well, such as MPELU [19]. Overall, there are various different alternatives to popular activation functions with advantages and drawbacks.

We note that there are many different types of activation functions, as mentioned above, and to consider all within a single framework is a difficult task. For this reason, we

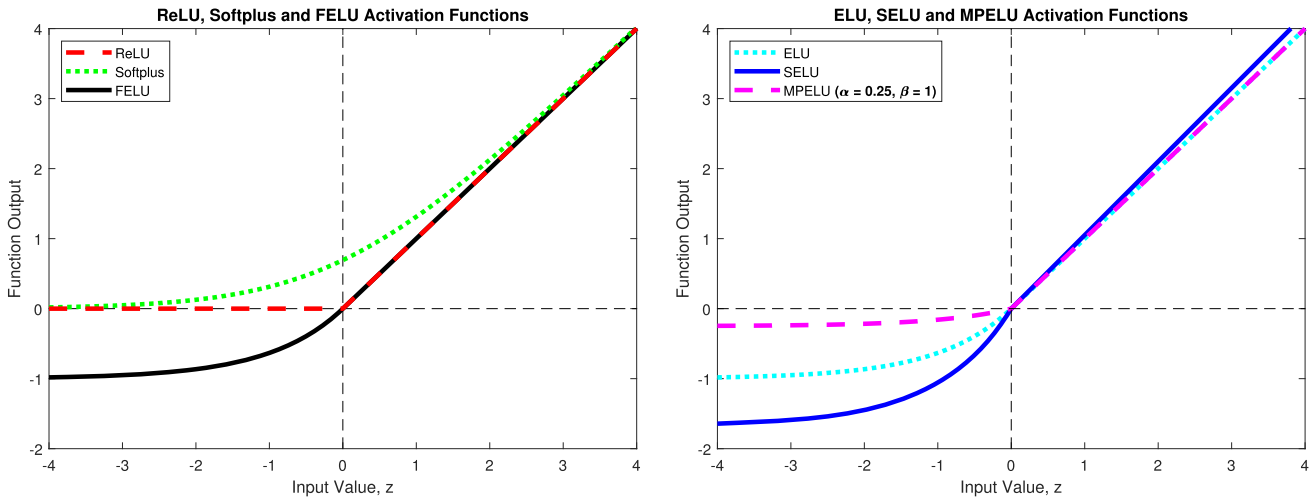

Fig. 1 ReLU, Softplus, FELU, ELU, SELU and MPELU activation functions are given

focus on rectifier-based activation functions in this paper, particularly on ReLU, ELU and use their variants such as MPELU [19], SELU [15], FELU [13] and Softplus [5], since these ones are popular, suggested and mostly utilized in deep learning applications. This work suggests another activation function, named as "ExtendeD Exponential Linear Unit" (DELU) as a replacement to ELU in the ReLU family of neural activation functions. Based on the advantages of rectifier-based activation functions and building upon their beneficial additions, DELU appears to be a viable alternative in comparison with existing activation functions in the literature and may perform better in some applications through careful selection of its parameters. Specifically, the behaviors of ReLU and ELU activation functions around the zero point are seen as subject to improvement. In this manner, DELU is designed as a modification to these parts of existing activation functions, ReLU and ELU. From other motivations inherited to DELU, overcoming the 'dead zone' problem of ReLU and seeking a better activation function than ELU are important to mention.

With these motivations, our work aims to make the following contributions:

- Introducing a novel activation function,
- Comparing it with rectifier-based activation functions,
- Presenting both classification and image segmentation examples.
- Using statistical methods in addition to tabulated results,
- Making a compatibility analysis by different initialization configurations,
- Optimizing the parameters of this novel activation function.

This paper is organized as follows. In the following section, we introduce the related work. Then, we propose our activation function DELU with heuristical and theoretical explanations. Afterward, we present the experiments conducted with ReLU, ELU, SELU, MPELU, FELU, Softplus and DELU for Fashion MNIST and CIFAR datasets using ResNet architecture. Then, we present our results for image segmentation networks with those activation functions. Then in the Sect. 5, we comment on the obtained results in detail with compatibility analysis, consider a hypothesis testing problem, utilize Z-score analysis and explore different parameters of DELU activation function. Finally, we give some concluding remarks at the Sect. 6.

2 Related work

The number of scientific works focusing on newly defined activation functions continually increases in recent years, see [11]. For more information, interested reader is referred to recent survey papers [5, 11] and references therein.

Since there are numerous activation function candidates, it would not be possible to discuss even the majority of them within the scope of our work. Hence, we will mainly focus on the ones which are mostly utilized in neural network studies and are most relevant to our work.

2.1 Classical activation functions

Especially in early neural networks, biologically inspired activation functions were used frequently. An important one is sigmoid, and its formula is given:

$$Sigmoid(z) = \frac{1}{1+e^{-z}} \tag{1}$$

This function is still used in recurrent neural networks, such as LSTM cells. Hard-limiter function is also one of these activation functions, given as follows:

$$hardlim(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0. \end{cases} \tag{2}$$

Finally, hyperbolic tangent ($tanh(.)$) is another classical activation function. Its definition is given below. Due to its zero mean, it is used in a wider context than the sigmoid in many neural network structures.

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{3}$$

2.2 Swish and its variants

Swish activation function, as given below, is a modification of sigmoidal given by (1) and is introduced in recent years and given below. It aims to benefit from the changing parameter β for better learning in neural network structures [11].

$$Swish(z) = z \cdot Sigmoid(\beta z) \tag{4}$$

We can point out its similarity with ReLU activation function as $\beta \to \infty$. Its well-known variants, such as Shape Autotune Activation Function [17] and Mish activation function, are also used in the literature.

2.3 Rectifier-based activation functions

First rectifier-based activation function is the ReLU activation function denoted here as $\mathrm{ReLU}(\cdot) : \mathbf{R} \to \mathbf{R}$ and given as follows:

$$\mathrm{ReLU}(z) = max\{0, z\} = \begin{cases} z & z > 0 \\ 0 & z \leq 0. \end{cases} \tag{5}$$

Various modifications of ReLU have been proposed in the literature. Among those, possibly the most important one is ELU activation function, which is first proposed in [12] and here denoted as $\mathrm{ELU}(\cdot) : \mathbf{R} \to \mathbf{R}$, is given as follows:

$$\mathrm{ELU}(z) = \begin{cases} z & z > 0 \\ e^z - 1 & z \leq 0. \end{cases} \tag{6}$$

For visual motivation, graphical representation of those functions is given in Fig. 1. Note that both in ReLU and ELU activation functions, the behavior is simply linear for $z \geq 0$. On the other hand, $\mathrm{ReLU}(z) = 0$ for $z \leq 0$ has a certain disadvantage, known as 'Dying ReLU' phenomenon where it stops learning [21]. Note that this is the main motivation behind the formulation of $ELU(z)$ for $z \leq 0$ as given in (6).

We used other activation functions in our comparisons as well. Specifically Softplus [5], SELU [15], MPELU [19] and FELU [13] are used in our comparisons. They are presented below with figures and formulas. For SELU, $(s, a) = (1.05, 1.673)$ are used. For MPELU, the most successful parameter pair for CIFAR-10 is used, $(\alpha, \beta) = (0.25, 1)$ from the work [19] with implementation [20]. FELU is a reformulation of same function from ELU and uses $2^{f(z)}$ in the multiplication instead of $e^{f(z)}$ in favor of faster execution times. Note that there is a small difference between them, due to approximating $ln2$.

$$\mathrm{Softplus}(z) = ln(1 + e^z) \tag{7}$$

$$\mathrm{FELU}(z) = \begin{cases} z & z > 0 \\ 2^{z/ln2} - 1 & z \leq 0. \end{cases} \tag{8}$$

$$\mathrm{SELU}(z) = \begin{cases} s \cdot z & z > 0 \\ s \cdot a \cdot (e^z - 1) & z \leq 0. \end{cases} \tag{9}$$

$$\mathrm{MPELU}(z) = \begin{cases} z & z > 0 \\ \alpha\left(e^{\beta z} - 1\right) & z \leq 0. \end{cases} \tag{10}$$

3 The proposed activation function

3.1 Function derivation and definition

We propose the following activation function:

$$\mathrm{DELU}(z) = \begin{cases} z & z > z_c \\ (e^{\alpha z} - 1)/\beta & z \leq z_c \end{cases} \tag{11}$$

where $\alpha \geq 0$, $\beta \geq 0$ and $z_c \geq 0$ are appropriate parameters to be determined. If we impose continuity on $DELU(.)$ at $z = z_c$, the following equation should be satisfied:

$$\beta z_c = e^{\alpha z_c} - 1, z_c \geq 0. \tag{12}$$

Note that $z_c = 0$ is always a solution of (12) for any $\alpha \geq 0$ and $\beta \geq 0$. The relation of DELU with ELU and ReLU is clarified in the following remark.

Remark 1 Note that if we choose $\alpha = \beta = 1$ and $z_c = 0$, (11) becomes the same as ELU given by (6). On the other hand if we choose $z_c = 0$, for $\beta \to \infty$ it follows that $DELU(z) \to 0$ for $z \leq 0$. Hence ReLU given by (5) could be considered as a limiting case of DELU given by (11). Note that a similar property also holds for Swish given by (4) as $\beta \to \infty$. So, by choosing appropriate values of $\alpha \geq 0$ and $\beta \geq 0$, we expect to have similar properties that ELU or ReLU offers. □

Note that, as explained above, $z_c = 0$ is always a solution of (12) for any $\alpha \geq 0$ and $\beta \geq 0$. However, for appropriate selection of the parameters α and β, we may have another solution of (12) with $z_c > 0$. This point is clarified as follows:

Fact 1: For any $\alpha \geq \beta > 0$, the only solution of (12) is $z_c = 0$. For $\beta > \alpha > 0$, (12) has a second solution $z_c > 0$. □

Proof Note that the left-hand side of (12) $L(z) = \beta z$ is linear in z with $L(0) = 0$ and $L'(z) = \beta$, where a superscript $'$ here and in the following denotes derivative with respect to z. The right-hand side of (12) $R(z) = e^{\alpha z} - 1$ is a strictly increasing function with $R(0) = 0$ and $R'(z) = \alpha e^z \geq \alpha$ for $z \geq 0$. For $\alpha \geq \beta > 0$, it is clear that the unique solution of (12) is $z_c = 0$. On the other hand, when $\beta > \alpha > 0$, since $L(z) < R(z)$ for sufficiently small $z > 0$. Since exponential functions grow faster than linear functions, there necessarily exists another (and unique) $z_c > 0$ such that (12) holds. □

Remark 2 Note that $\alpha \geq 0$, $\beta \geq 0$ and $z_c \geq 0$ are considered as the parameters of DELU. Since we impose the continuity of (11) at $z = z_c$, (12) should be satisfied. Hence if we choose any two of these parameters, the remaining parameter could be found from (12). For example, if we choose $z_c = 1$ and $\alpha = 1$, then (12) yields $\beta = e - 1$. Hence then, the proposed model offers some flexibility for parameter tuning to obtain better results in neural network applications, which will be explored in the Analysis section. For practicality, we should start with a selected specific parameter set for (11) in our work and use it for most of the experiments. Note that for $\alpha = 1$, $\beta = 2$, from (12) we obtain $z_c = 1.25643$. We note that this parameter set will be utilized throughout the paper except Sect. 5.3, "Optimizing the Parameters of DELU." □

The graphical representation of (11) with the parameter set given by (12) is given in Figure 2a. DELU function

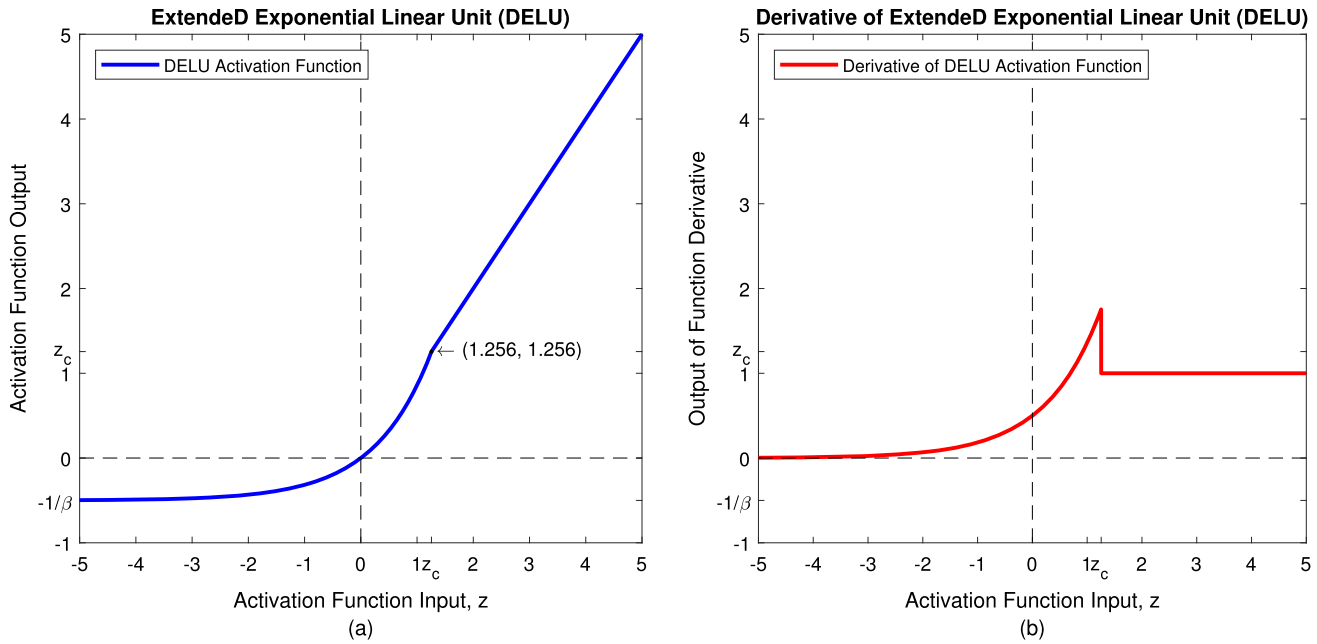

Fig. 2 Graphs of DELU activation function (**a**) and its derivative (**b**) for $\alpha = 1, \beta = 2$

given by (11) is differentiable except the point z_c, and its derivative is given in (13). Its graphical representation with the default parameters as indicated in Remark 2 is given in Fig. 2b.

$$\text{DELU}'(z) = \begin{cases} 1 & z > z_c \\ \left(\frac{\alpha}{\beta}\right) e^{\alpha z} & z \leq z_c. \end{cases} \tag{13}$$

Clearly derivative is discontinuous at $z = z_c$ unless $\alpha = \beta = 1$ and $z_c = 0$, which is the ELU case given by (6). Note that many variants of ReLU also have the same discontinuity problem. However, this does not pose a problem for applying the backpropagation algorithm for DELU, like those variants. In addition, the continuous values of the derivative of DELU around zero may affect the training process; this point will be explained in the next section.

3.2 Effects of DELU in training process

DELU activation function as given by (11) has certain differences as compared with ReLU and ELU given by (5) and (6), respectively. The main difference is on the behavior of the activation functions around $z = 0$, especially for $z \geq 0$. While the slope of both ReLU and ELU for $z \geq 0$ is equal to 1, see (5) and (6), the slope of DELU activation function changes depending on z. To elaborate further, let us consider the DELU and its derivative as given by (11) and (13), respectively. For future reference, we will rewrite these equations with the parameters $\alpha = 1$, $\beta = 2$ and $z_c = 1.25643$, which will be our default parameter set to be used in the rest of the paper, as explained in Remark 2.

As can be seen from Fig. 2b, while $\text{DELU}'(0) < 1$, we have $\text{DELU}'(z) \geq 1$ in the range $z \in [z_x, z_c)$ for some $z_x > 0$. From (13) with parameters $\alpha = 1$ and $\beta = 2$, it easily follows that $z_x = \ln 2 = 0.69314$. We may consider the effect of this slope change on the early stages of the training process. Usually, the weights of the neural network are initialized randomly and typically assigned by using a Gaussian probability distribution with zero mean. Since input of each neuron is a weighted sum of certain signals in the neural network, it is natural to expect the input of each neuron initially as a Gaussian random variable with zero mean. If the weights are updated according to standard backpropagation-type minimization algorithm, then weight updates will be proportional to the derivative of the activation function around $z = 0$. At this point, behavior of DELU will be slightly different than the behavior of ReLU and ELU, since $\text{DELU}'(0) = 0.5 < 1$ and for others we have $\text{ELU}'(0) = \textit{ReLU}'(0) = 1$. Hence we expect that, initially, weights will be updated slowly when DELU is used, which may prevent some neurons from early saturation, while in ReLU and ELU cases the weights will be updated by a larger amount which may force some neurons to early saturation. Intuitively, we may expect that in the early stages of training while usage of DELU offers a larger set of search directions for weight update, hence a better chance for loss minimization as compared to ReLU and ELU. The process alluded to above holds as long as $z \in [0, z_x)$. As the training evolves, we may expect that the inputs to neurons increase as well. When $z \in [z_x, z_c)$, we

have $\text{DELU}'(z) > 1$; hence, in this case we expect a faster weight update as compared to ReLU and ELU. Hence by combining these arguments, we expect that the usage of DELU offers a slow weight update at the early stages of the training but fast weight updates at later stages, which intuitively may result in better loss minimization. This argument is the main reason for choosing the activation function of DELU as given by (11).

The arguments given above can also be analyzed by using suitably defined random variables, see, e.g., [17]. To elaborate further, as alluded in the argument given above, assume that the initial neuron weights are assigned randomly and let the weights be initialized by using a Gaussian distribution or any continuous distribution centered around zero. Here, the only requirement needed is an independent and identical distribution for all neuron weight parameters, which is conventionally done in neural network initialization. In these cases, since the input z of any neuron is a weighted linear sum of such random selections, by using Central Limit Theorem [22] it is reasonable to assume that the input z of each neuron may also be associated with a random variable Z with a Gaussian distribution $N(0, \sigma^2)$, whose probability density function (PDF) is given as:

$$f_Z(z) = \frac{e^{-z^2/2\sigma^2}}{\sigma\sqrt{2\pi}} \tag{14}$$

As a result, if the activation function is $DELU(z)$, then output of the neuron is also a random variable, which is given as:

$$\text{DELU}(Z) = \begin{cases} Z & bZ > 1.25643 \\ (e^Z - 1)/2 & bZ \leq 1.25643. \end{cases} \tag{15}$$

The expected value and variance of the output $DELU(Z)$ can be given as:

$$E(\text{DELU}(Z)) = \int_{-\infty}^{\infty} DELU(Z) f_Z(z) dz \tag{16}$$

$$Var(\text{DELU}(Z)) = E(DELU(Z)^2) - (E(DELU(Z))^2 \tag{17}$$

where the operators $E(\cdot)$ and $Var(\cdot)$ refer to expected value and variance of a random variable, respectively. It follows from (11) and (13) that these values are well defined, see, e.g., [17].

Using their PDFs, we calculated the mean values for only ReLU, ELU and DELU activation function choices, as they are the best ones in our CIFAR-10 comparison and for the sake of simplicity, and the summarizing figure of their mean values as a function of input standard deviation is given. Comparative results for standard deviation (which is defined as square root of variance) and mean value (μ) functions for ReLU, ELU and DELU are given in Fig. 3. At the beginning of the training process, DELU activation function outputs the least mean, while ReLU and ELU produce more. This corresponds to the slow allocation of neurons to drop the possibility of early over-fitting, heuristically. Smaller mean values at the beginning of training process are also preferred in well-known works such as Batch Normalization [23]. After the training speeds up and features are determined in the neural network as parameter values, the standard deviation of weights tends to increase, as pointed out in [23]. So, as the standard deviation of input Gaussian PDF increases, mean value of the output random variable increases as well. Here, DELU will give higher mean values, which will speed up the learning process by increasing derivative values in comparison with ReLU and ELU, and training with DELU might be better.

The arguments given above, as well as Fig. 3, could be considered as an alternative confirmation which justifies our earlier comments on the first slower then fast weight updates, given at the beginning of this section.

In addition to the argument given above, another comment can be made. Due to Gaussian nature and low variance values of the inputs of activation function, the change caused by DELU activation function affects more than it may seem at first glance. Many neurons are initially within the neighborhood of origin and in this vicinity, and derivative values of DELU are different than ReLU and ELU as illustrated in previous sections; making the learning slower at the beginning and faster at late stages of training—with higher derivative values as well. Keeping the relations of error gradient with activation function and derivative of activation function in mind, the

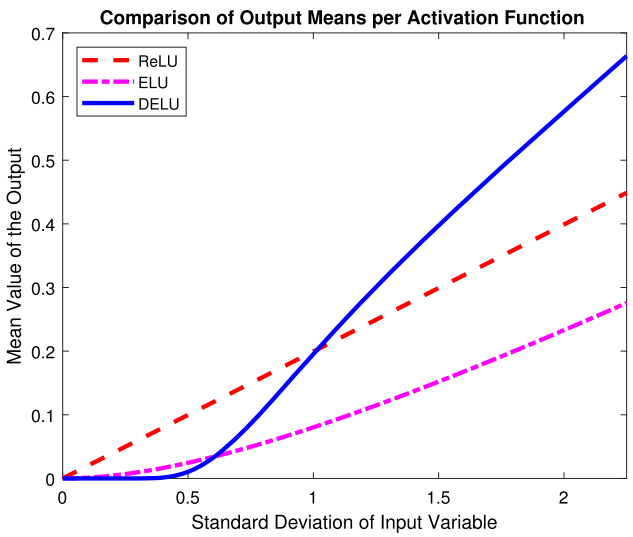

Fig. 3 Obtained relation graph for ReLU, ELU and DELU activation functions

aforementioned impact of activation function choice on the learning process may be significant. Therefore, with explanation about PDF functions complete, it is important to mention that DELU brings a different behavior compared to ReLU and ELU.

3.3 Output landscape comparison

While considering the arguments given above, it is also important to give experimental evidence about our statements on DELU. To this end, we add output landscape comparison. Output landscape is created by using a randomly initialized, multilayer neural network. We give 2 input values as x–y couples from the 2D plane and get a single output value for each input couple, which is mapped into another 2D plane. It is centered at origin for all activation functions. Few examples are given in Fig. 4 for ReLU, ELU and DELU.

In [17], output landscapes and their comparative smoothness are proposed as indicators for finding better neural network activations. Having a smoother output visual is described to lead to smoother loss surfaces during the training process in [24]. For the purpose of strengthening our claim that DELU activation function offers a larger set of search directions for weight update, hence a better chance for loss minimization, we included the output landscape figures in Fig. 4. Note that our comparison contains ReLU, ELU and DELU only as they are the best three activation functions on CIFAR-10 in our work.

As seen in the different output landscapes, different results are obtained for ReLU, ELU and DELU activation functions. Vertical and horizontal axes of outputs correspond to values of x–y input couples, while output values increase from navy blue to yellow color (darker to lighter). As discussed in [17], ReLU have sharper parts in comparison with ELU and DELU. On the other hand, DELU has the smoothest edge in the middle, while ELU has the least sharp for the top left corner in comparison with both ReLU and partially, DELU. The smoothness in the middle section of the output landscape of DELU may match with the heuristical explanation given above; not falling over sharp edges and allocating themselves to the features at early training stages. However, it is not entirely possible to champion one activation function over the other with certainty, as the output landscapes are generated by using randomly initialized 5-layer feedforward neural networks. It is only a possible clue that DELU, with a smoother central area, can make a positive difference for learning process.

4 Experiments

In our simulations, we use default definitions of ReLU, ELU, SELU, Softplus, FELU and $(\alpha, \beta) = (0.25, 1)$ values for MPELU. For DELU, we use (11) with default values.

4.1 Image classification

4.1.1 CIFAR-10 and CIFAR-100 dataset networks

For comparing the activation functions, multiple datasets are classified with the well-known and reliable ResNet (Residual Networks) architecture with practical Keras library, based on TensorFlow implementation, while the only difference was the activation functions we used for a fair comparison. Our aim in this simulation analysis is to observe the effect of various activation functions for a given network structure and to observe if one particular selection results in consistent improvement in performance. Specifically, for each activation function, 10 networks from different types of ResNet family are trained for CIFAR-10, CIFAR-100 and Fashion MNIST datasets [25, 26]. Test results are used to compare the activation function successes, both by comparison using success tables and statistical methods for the case of CIFAR datasets (Fig. 5).

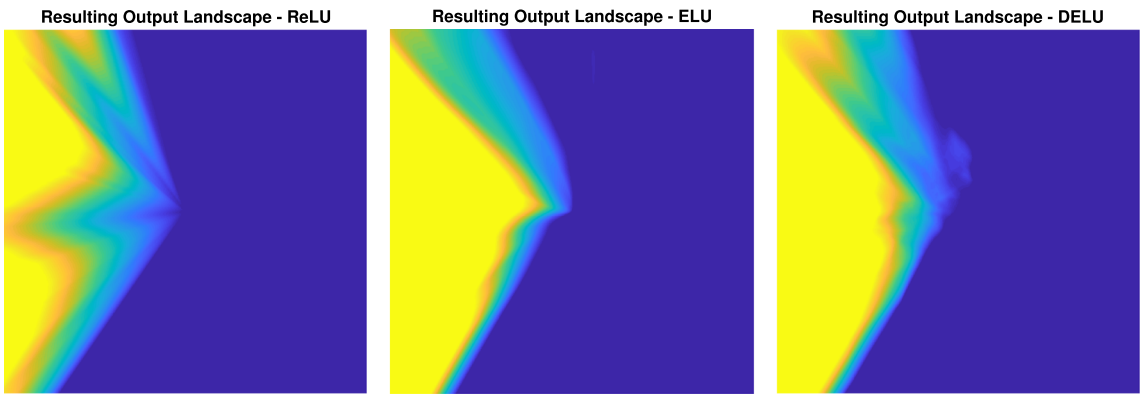

Fig. 4 Obtained landscape graphs for three different activation functions

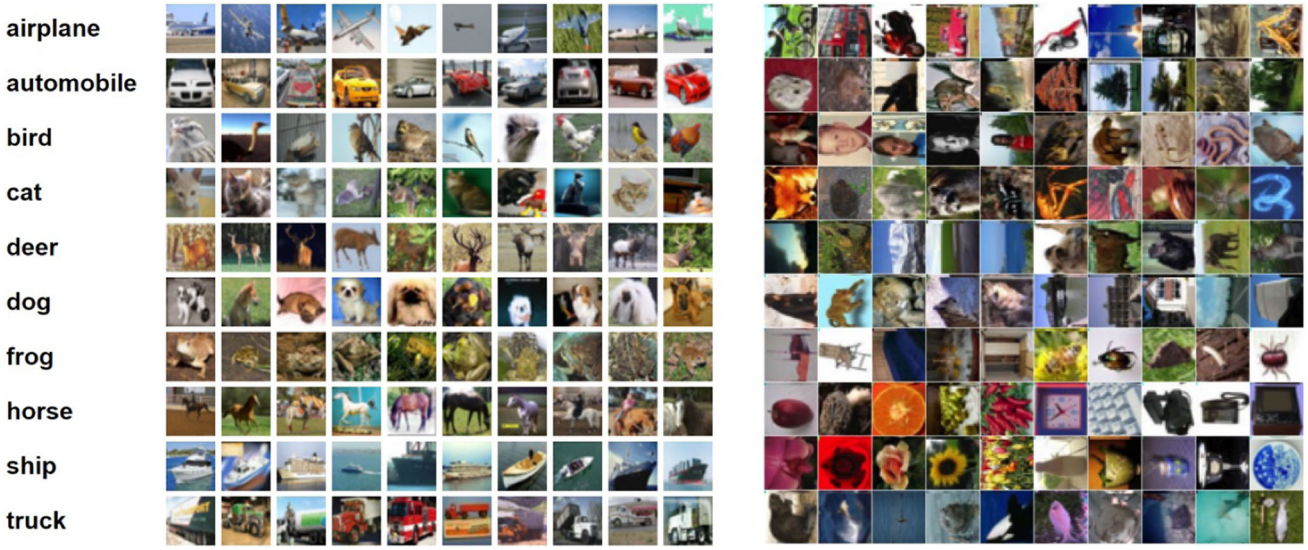

Fig. 5 Example data samples: CIFAR-10 at left and CIFAR-100 at right [25]

CIFAR-10 and CIFAR-100 datasets are popular datasets in classification benchmark tasks, and they are still in common use [25]. They consist of colored (RGB) 32 × 32 pixel images of different classes, 10 for CIFAR-10 and 100 for CIFAR-100 in fine label mode. By using results of these datasets on original ResNet architecture with different activation functions we will show the success rates and try to highlight the differences of results, using statistical methods.

Mainly, two types of the ResNet architecture as displayed in Fig. 6 are used in this work: ResNet20-v1 for CIFAR-10 [27] and ResNet164-v2 for CIFAR-100, both are constructed using various blocks illustrated in work [28]. Both network architectures are thoroughly described in [29]. The default code architecture is taken from source [30], and it is designed for CIFAR-10 dataset using Keras library of TensorFlow, Python. By slightly increasing the input and output class sizes of the network in a regular way, we have adapted ResNet164-v2 for CIFAR-100 dataset (fine-mode). In total, 2 types of ResNet architectures are used for this article.

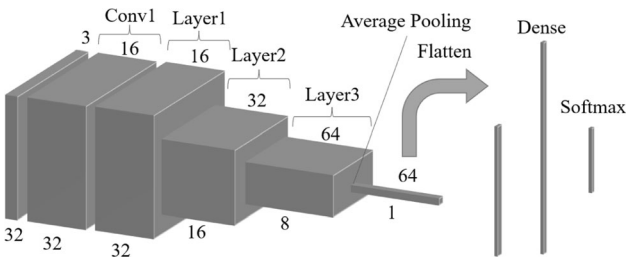

Fig. 6 General ResNet structure is shown using blocks, inspired from work [28]

Default options for neural networks are used apart from the choice of activation functions (except Softmax outputs). Networks are initialized with He Normal [31] initialization and kept the same for all activation functions for fairness—including SELU. Adam learning algorithm is employed for 200 epochs and categorical cross-entropy loss function is used, while default data augmentation is kept. L2 regularization with parameter 0.001 is also employed for training with 45,000 samples, where first 5000 are left for validation. Thus, it is possible to compare our results for different activation function choices.

Results of our simulations are given in Tables 1 and 2. Test results of 10 networks obtained in 10,000 samples are compared for every activation function.

As seen in Tables 1 and 2, DELU leads in both sets of experiments for mean success rates. However, its standard deviation is not small, especially in CIFAR-100. Afterward ReLU comes, then ELU is third in CIFAR-10. In the case of CIFAR-100, MPELU is second and ReLU is third, while DELU is in the lead with a slight margin. In CIFAR-10, ELU has the highest minimum and maximum success rates, while in CIFAR-100, DELU has the highest maximum and MPELU has the highest minimum success rate.

4.1.2 Fashion MNIST dataset networks

To enhance our findings, we decided to adapt one of the ResNet networks for the Fashion MNIST dataset as explained in the previous section. This dataset is given in [26] as a replacement for MNIST which is already overpopulated with 99.5% and higher success rate networks. The researchers designed this dataset as an alternative to MNIST, and because of that reason 28 × 28 pixel size and

Table 1 CIFAR-10, 10 ResNet20-v1 networks

	Success rate percentages			
	Minimum (%)	Mean (%)	Maximum (%)	Standard deviation
ReLU	89.67	90.099	90.40	0.2372
ELU	90.11	90.361	90.77	0.1976
DELU	90.07	90.468	90.74	0.2492
SELU	88.37	88.965	89.34	0.2915
MPELU	89.17	89.603	90.27	0.3453
FELU	89.29	89.566	89.72	0.1466
Softplus	88.64	89.009	89.55	0.3028

Table 2 CIFAR-100, 10 ResNet164-v2 networks

	Success rate percentages			
	Minimum (%)	Mean (%)	Maximum (%)	Standard deviation
ReLU	73.11	73.406	73.81	0.2223
ELU	72.06	72.537	72.92	0.3335
DELU	72.82	73.509	74.19	0.4157
SELU	71.38	71.794	72.23	0.2476
MPELU	73.15	73.485	73.95	0.2217
FELU	71.77	72.347	73.02	0.1466
Softplus	72.00	72.676	73.47	0.4158

grayscale specialties are kept the same. Instead of digits, different accessories and clothes are used as 10 item classes of the dataset (Fig. 7).

We used this dataset with a slight modification for not changing the ResNet20-v1 architecture. Instead of 28x28 pixel size, we added enough black pixels at top and left of the image to meet the 32×32 standard. For all color channels of RGB, we copied the content of grayscale into all there channels, making the sample size $32 \times 32 \times 3$ - same with CIFAR samples. Thus, it became possible to use these samples with ResNet architecture coded for Keras library.

In Table 3, there are 10 networks trained for each activation function with 5000 validation samples, echoing the previous configuration. Success rates are obtained for each function as seen here, for 200 epochs training. For the results given in Table 4, we only use the network with highest validation success rate obtained during 200 epochs among 10 networks for each activation function.

As seen in Tables 3, 4, we can see different results in each table separately. However, DELU is ahead with 93.903% mean success rate, in both of the tables. MPELU became the second best activation function in these success rates. Indeed, DELU also shares the lead in minimum success rates among both tables with MPELU, and this time MPELU leads in maximum success rates for all Fashion MNIST results obtained in this section. DELU's

Table 3 Fashion Mnist, 10 ResNet20-v1 networks at the end of 200 epochs

	Success rate percentages		
	Minimum (%)	Mean (%)	Maximum (%)
ReLU	93.42	93.708	93.87
ELU	93.43	93.766	94.07
DELU	93.41	93.784	94.10
SELU	93.43	93.681	94.03
FELU	93.57	93.747	93.84
MPELU	93.66	93.845	94.17
Softplus	93.35	93.713	94.05

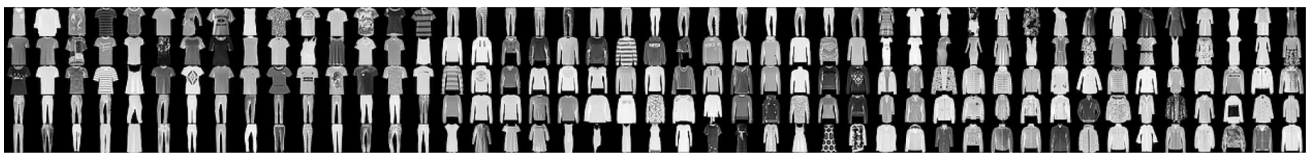

Fig. 7 Fashion MNIST dataset samples taken from [26]

Table 4 Fashion Mnist, 10 ResNet20-v1 networks with highest validation success

	Success rate percentages		
	Minimum (%)	Mean (%)	Maximum (%)
ReLU	93.53	93.694	93.94
ELU	93.44	93.757	94.02
DELU	93.70	93.903	94.11
SELU	93.37	93.565	93.74
FELU	93.50	93.749	93.92
MPELU	93.70	93.870	94.14
Softplus	93.12	93.739	94.14

mean success rate, 93.903%, makes this result pretty good for this dataset in the existent literature—without any further configuration changes and in a network with 0.27 million parameters only—while extra data were not used in training as well.

For both network results, default data preprocessing and augmentation is left unchanged, and no additional fine-tuning is applied. In result, this additional experiment set and simple comparison between activation functions also showed that the success of DELU is not limited to CIFAR datasets, but instead it is an activation function that can perform better in different datasets as well.

4.2 Image classification

4.2.1 Oxford IIIT pets dataset networks

To further enhance our results, we decided to add image segmentation task, in addition to image classification, to our work. Specifically, we used Oxford IIIT-Pets dataset which contains different breeds of dog and cats visuals [32]. Dataset contains colored images with varying sizes and nearly equally splitted with 3680 training and 3669 test samples. Researchers included trimaps and annotations for determining the success rates and provided their benchmark success for Intersection over Union (IoU) as 65% for the image segmentation task. An example is given below, alongside the formula of the IoU scores. Black pixels correspond to certain pixels of animal, white pixels stand for ambiguous region and gray pixels are certain background, in the ground truth images (Fig. 8).

The network used in this task is different than ResNet. We used this dataset with a different network, taking a new structure from Keras library's work "Image segmentation with a U-Net-like architecture" [33] which is given in Fig. 9. Here, network architecture is mainly kept same and official division of training and test samples is respected in

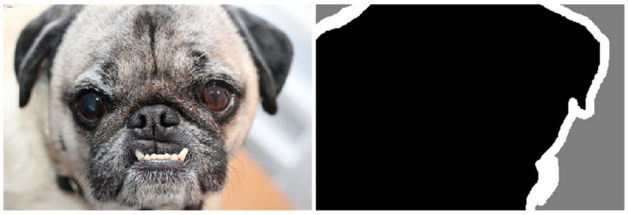

Fig. 8 Oxford IIIT-Pets data sample and its truth map, constructed from dataset [32]

our application. Sparse categorical cross-entropy is employed as the loss function with RMSprop learning algorithm. The output has the original size of the input, to check with the trimap segmentation ground truth for verification purposes. In total, 41 samples from training set are put aside for validation purposes in the training process. Accuracy of the image segmentation, using overlapped black and white pixels, is defined as:

$$\text{Accuracy}\,(\%) = \frac{\text{Overlapping pixels of truth map and our guess}}{\text{Amount of black \& white pixels in ground truth}} \tag{18}$$

The success rates of trained image segmentation networks are calculated with a specific formula given in (18), for each sample in the dataset. In this way, ambiguous region for photographed animal is also added to the certain pixel region that differentiates the animal from the background.

The main change is the internal activation functions employed, and each activation function is tried again separately for a fair comparison. In this manner, training for 300 epochs with a mini-batch size of 1 and 320×320 colored pixels input size, this time 20 independent and identical neural network results are obtained. For each activation function, minimum, mean and maximum success rates are presented in Table 5. As DELU takes a slight lead in mean success rates, we can claim these results show that it is possible to claim that DELU can be better than others in the image segmentation networks as well.

Success rates are close to original authors' benchmark of 65%; therefore, these rates are still remarkable for our comparison reasons. It can be possible to test DELU with different setups and success rates, to see whether it is suitable for such tasks and configurations. This point can be investigated further.

Before concluding the Sect. 4, it is important to mention the trade-off between performance and time constraints for this new activation function in comparison with the other activation functions used in this work. DELU requires more time for training in contrast to the others; however, this increase is not large. Around 15% increase is observed for the training time of Unet architectures used in this work

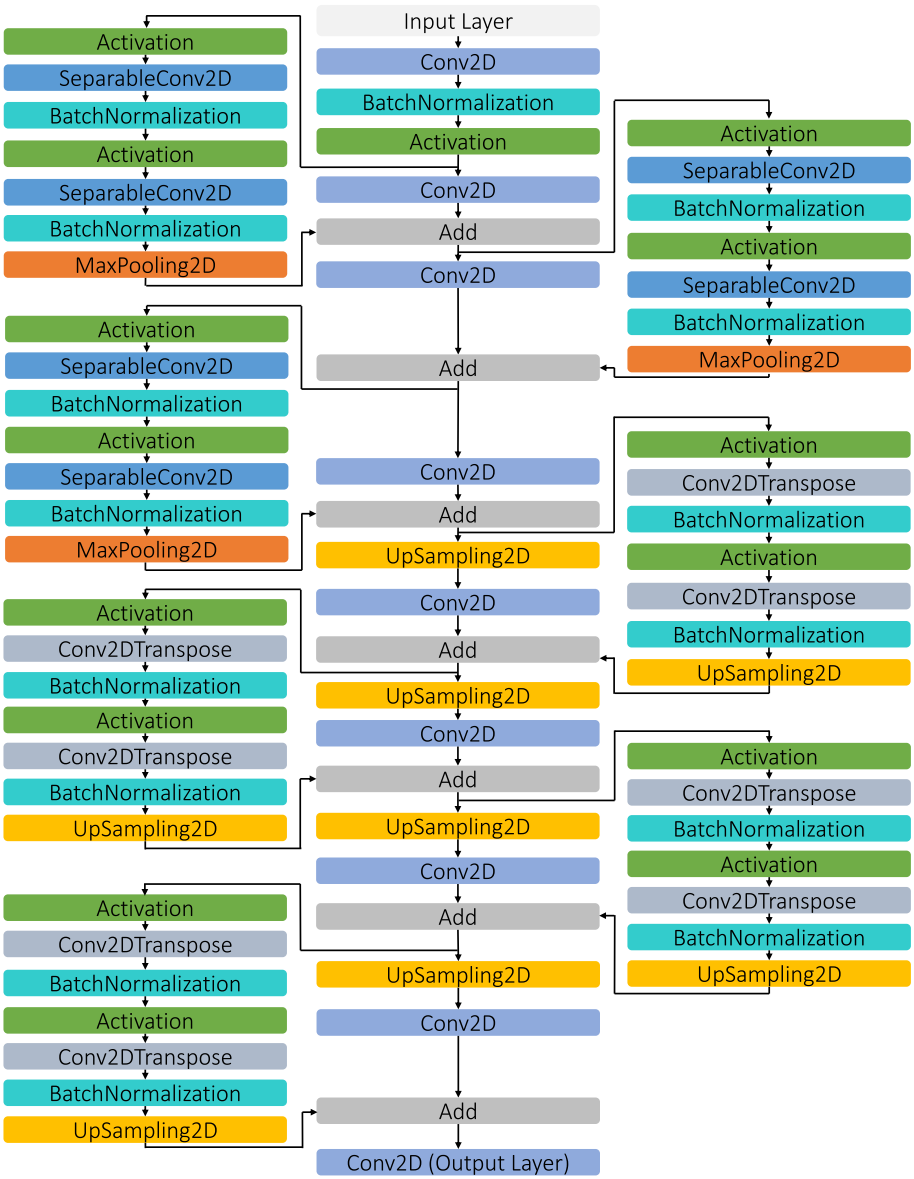

Fig. 9 Representation of our Unet, takes a image input and outputs segmented image

Table 5 Oxford-IIIT pets dataset, 20 U-Net-Like networks

	Success rate percentages			
	Minimum (%)	Mean (%)	Maximum (%)	Standard deviation
ReLU	59.272	61.251	62.703	0.898
ELU	58.920	60.911	62.611	0.975
DELU	57.269	61.299	62.769	1.146
SELU	58.655	60.891	62.189	1.019
MPELU	58.944	60.968	63.226	1.106
FELU	59.439	61.172	62.620	0.820
Softplus	38.213	59.442	63.058	5.210

with 300 epoch training settings, while the inference time for 3669 test samples is nearly the same with a slight increase as seen in Table 6. This is not surprising because of the effect of the DELU in the backpropagation algorithm. On the other hand, increases obtained in the mean success rates are making this trade-off worthy as the results imply.

Table 6 Mean times of classification and segmentation networks in seconds

	F. MNIST		CIFAR-10		CIFAR-100		Oxford Pets	
	Train	Test	Train	Test	Train	Test	Train	Test
ReLU	5252.4	2.658	4348.6	2.404	48,915.7	20.578	25,864.1	11,015.9
ELU	5214.8	2.654	4338.2	2.401	49,100.6	20.425	25,803.2	11,026.0
DELU	6153.9	3.053	5127.7	2.931	63,551.5	30.251	30,062.4	11,075.0
SELU	4159.2	3.132	3428.6	2.977	42,639.9	34.535	28,513.3	10,944.3
MPELU	4168.6	2.760	3453.4	2.647	43,977.5	25.124	28,775.8	10,816.7
FELU	4267.8	3.146	3572.5	2.985	47,986.2	30.788	30,018.7	10,893.9
Softplus	5513.3	2.701	5466.2	2.410	52,288.8	20.844	25,681.1	11,097.5

For networks with both CIFAR datasets and Fashion MNIST dataset, training and inference times are measured and inference times are tabulated as seen in Table 6. However, due to use of GPUs and CPUs under different loads in the training and inference of the networks in the Fashion MNIST and CIFAR datasets, it is not entirely fair to make a decision by using their timings only. However, in general DELU needs more than 15% additional training and inference time in comparison with ReLU, ELU and SELU, and DELU needs 2–3% less training time than MPELU for Fashion MNIST and CIFAR datasets in general. Compared to Softplus, DELU is around 10% slower in Fashion MNIST networks and faster around 7% in CIFAR-10 dataset specifically. In the light of those results, it can be said that the training time of DELU activation function is slightly higher than the ReLU and ELU. However, since the training is an offline process and performed only once, but the resulting network structure afterward utilized mainly for testing, one may argue that the inference time is more important. Indeed, many training methods actually aim at reducing inference time, see, e.g., [34, 35]. Since the inference time of DELU is comparable to the other rectifier alternatives, a bit longer training could be considered as acceptable, considering the slight increase we have in the success rates.

5 Analysis

In this section, we present hypothesis testing and Z-score analysis for the results obtained in CIFAR datasets, compatibility analysis with different initializations and exploring DELU alternatives with different (α, β) values.

5.1 Compatibility analysis

Here, the results of activation functions are obtained with different initializations to show the compatibility and insensitivity of DELU to change of initialization methods. CIFAR-10 networks are used for this purpose, and we tried Uniform and Gaussian (Normal) instead of default He Normal initialization, with same ResNet20-v1 structure. Training settings are also kept the same, and resulting success rates are given below for minimum, mean and maximum success rates obtained in 200 epoch-long 10 runs per each activation function.

As seen from Table 7, change of initialization methods does not change the leading activation function. DELU obtained the highest success rate in both uniform and Gaussian initialization, while ELU and ReLU follow in minimum, mean and maximum success rates among all runs (with a minor exception of equality at minimums, Gaussian initialization case). Please note that those success rates are less than He Normal initialization results, especially at means.

In addition to CIFAR-10 networks, Fashion MNIST networks are also used for a similar experiment as well. By using network and training configuration the same as before and trying He and Glorot (Xavier) versions of uniform and Gaussian initializations, these three successful activation functions are compared again. Results are obtained from networks tested after 200 epochs of training exactly, without considering their validation success rates. The resulting success rates are presented in Table 8.

Here, we can see the DELU activation function obtaining the highest success rates for all sets of activation function-initialization combinations for mean success rates. In detail, DELU also has the best minimum success rate among all, while in this occasion, ELU obtains the highest success rate within the maximum success rates. For all the columns presented in Table 8, DELU has the highest success rate in 7 columns, while ELU leads in 5 columns and ReLU only leads 2 columns. Also, for the mean success rates, DELU leads 3 out of 4 columns, which is more important. With these results and DELU's success rate lead at hand, we may claim DELU's compatibility with different initializations.

5.2 Hypothesis testing and Z-score analysis

We assume that the success rates given above are related to the associated neural networks in a random manner. This is

Table 7 CIFAR-10, 10 ResNet20-v1 networks per each activation function-initialization set and their success rates

	Minimum successes		Mean successes		Maximum successes	
	Uniform (%)	Gaussian (%)	Uniform (%)	Gaussian (%)	Uniform (%)	Gaussian (%)
ReLU	89.560	89.760	89.916	90.025	90.400	90.370
ELU	89.270	89.450	89.958	89.970	90.230	90.440
DELU	89.710	89.760	90.078	90.396	90.460	90.970

Table 8 Fashion MNIST, 10 ResNet20-v1 networks per each activation function-initialization set and correspondent success rates

	Minimum successes		Mean successes		Maximum successes	
	Glorot (Xavier)		*Glorot (Xavier)*		*Glorot (Xavier)*	
	Uniform (%)	Gauss (%)	Uniform (%)	Gauss (%)	Uniform (%)	Gauss (%)
ReLU	93.516	93.240	93.752	93.740	94.000	94.110
ELU	93.640	93.550	93.913	93.879	94.040	94.270
DELU	93.570	93.570	93.942	93.762	94.130	94.000
	Minimum successes		Mean successes		Maximum successes	
	He		*He*		*He*	
	Uniform (%)	Gauss (%)	Uniform (%)	Gauss (%)	Uniform (%)	Gauss (%)
ReLU	93.420	93.420	93.622	93.708	93.850	93.870
ELU	93.680	93.430	93.883	93.766	94.100	94.070
DELU	93.760	93.410	93.889	93.783	94.060	94.100

a reasonable assumption, since during the training at least initial weights are generated randomly. Hence we assume that we can define random variables which generate these numbers, for each activation function. Let us call the random variable for success rates of ReLU as S_{ReLU}, which maps probabilities to the success rates from 0 to 100 continuously. This can be done for other activation functions as well, specifically S_{ELU} and S_{DELU} can be defined similarly. With these definitions, we can furthermore assume that all these random variables have an underlying Gaussian PDF, which will be shown in the next steps with minimal assumptions.

To assume that there is an underlying Gaussian PDF which explains the distributions of these rates for experiments, we need to use an approximation via well-known Central Limit Theorem (CLT) [36]. Also, to start from success rates obtained, we use a nonparametric kernel distribution [37]. To summarize, nonparametric kernel distribution is done by using n independent realizations from the random variable and expressing the PDF of a single random variable from specialties of those n realizations. We use Gaussian kernels for our case.

The codes provided in MATLAB documentaries are used for showing that this distribution evolves into Gaussian as expected by the CLT [38]. By using those assumptions and confirming that underlying PDF of random variable S_{ReLU} (likewise S_{ELU} and S_{DELU}) is Gaussian, it will be possible to apply hypothesis testing as well, which will be explained in the rest of this section.

Although the results give us a feeling that newly designed DELU might be better than the existent ones, it is still not enough to make a solid statement. For this purpose, we will use hypothesis testing from detection and estimation theory [36]. To use this test easily, we assume inherent probability distribution of presented results as Gaussian. To this end, from Tables 1 and 2, we use mean and standard deviation values from test results of 10 networks and use CLT, while assuming their inherent probability distribution is Gaussian [36]. Next we attempt to show that this assumption is valid, by using the procedure as given below. For this purpose, we will consider the results given in Table 1.

The process to obtain a Gaussian probability distribution is illustrated in Fig. 11. Technically, we perform the operations outlined below in MATLAB. Note that changing the standard deviation values will only shrink or expand the resulting Gaussians, but not the Gaussian shape in the limit. Resulting shape is given in Fig. 11 for ELU activation function success rate results. Looking at the resulting shapes, the CLT assumption of reaching to a Gaussian distribution after convolving the PDF with itself enough times holds.

- *Step* 1: To build a probability function, we need to use obtained samples from success tables. We can take the results of ReLU activation function as an example. For each one of 10 samples, we place a Gaussian function: Centered at this sample value with a standard deviation of $\mathrm{STD}_{\mathrm{ReLU}}$, which is the standard deviation of the set of all 10 samples. This will result in a function similar to a kernel distribution.
- *Step* 2: Repeat this procedure for ELU and DELU activation functions to find their corresponding kernel probability distributions. A typical application for ELU activation function is given in Fig. 10.
- *Step* 3: Then divide this resulting graph by 10 to make this function a valid PDF with its definite integral equal to 1, which was sum of 10 Gaussian PDFs and its definite integral value was 10.
- *Step* 4: As this is a nonparametric representative PDF function of S_{ReLU}, we can use it to check whether it will converge to a single Gaussian PDF after convolutions. This is expected by the Central Limit Theorem (CLT), as long as the summation of independent and identically distributed random variables corresponds to the convolution of their PDFs [36]. To finish this step, simply convolve this PDF with itself 9 times, making 10 functions convolved for the CLT assumption check. Repeat this procedure for ELU and DELU activation functions.
- *Step* 5: Normalize resulting Gaussian PDF for verification (use convolution with padding for preciseness).

A typical application of the steps outlined above for ELU and DELU activation functions by using the results given in Table 1 is given in Fig. 11. After applying this routine for both activation function results, it is clear that the shape converges into a nearly ideal Gaussian PDF as expected by the Central Limit Theorem for ReLU, ELU and DELU activation functions, as illustrated in Fig. 12. Thus, we have shown that the random variable, characterized by 10 success rates for each activation function, can be assumed to have a Gaussian PDF via Central Limit Theorem as illustrated above.

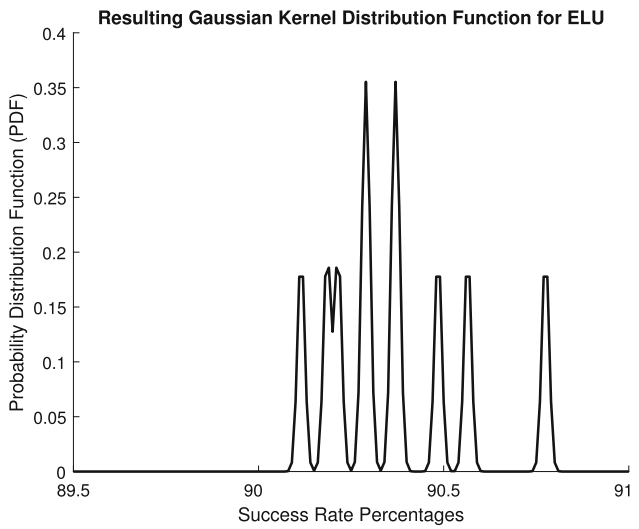

Fig. 10 The nonparametric PDF for random variable S_{ELU} of success rates, which will be used according to process described here

After we have the Gaussian shape for PDFs, we can also claim that mean and standard deviation values obtained from success rates can be used directly to place a Gaussian PDF for each one of activation functions. In our paper, this approach will be followed for simplicity and because of the closeness of mean values and evident Gaussian shapes obtained after convolution with Gaussian-kernel density PDFs. Therefore, standard deviation value is picked from the tables and divided by $\sqrt{10}$ (number of experiments in square root), for having the closest PDFs to original outcomes of experiments from random variables as complying with CLT assumptions, making both graphs look similar.

At Figure 12a, we can see the visualization of Gaussians for the explained comparison procedure of the activation functions DELU, ELU and ReLU for this case. Here we see that DELU has its mean success rate higher than others for CIFAR-10 and ResNet20-v1. However, the standard deviation of DELU results is higher than ReLU, as it seems by the wideness of its Gaussian in Figure 12a. Similar to success rates obtained in Table 2, DELU maintains its lead with a slight margin over ELU with representative Gaussians as well. At Figure 12b, we can see aforementioned assumptions realized for DELU, ReLU and less successful one, ELU. Here we see DELU has its mean success rate higher than others for CIFAR-100 and ResNet164-v2, similar to Figure 12b and Table 1. Confidence interval obtained from these figures is important, according to statistical significance value chosen for comparison reliability.

To investigate the statistical significance of these results, we resort to hypothesis testing methods which are typically utilized in decision making [39]. For this purpose, we consider two hypotheses relevant to our case. H0 is the null hypothesis where it implies that replacing ReLU and ELU with DELU does not have a significant effect on increasing success, claiming that the change is statistically insignificant. H1, on the other hand, is the hypothesis which implies that replacing ELU or ReLU with DELU does have a significant effect on increasing success. In short, the hypotheses to be tested are given below.

- H0: Replacing ReLU (or ELU) with DELU does not increase success rates
- H1: Replacing ReLU (or ELU) with DELU does increase success rates

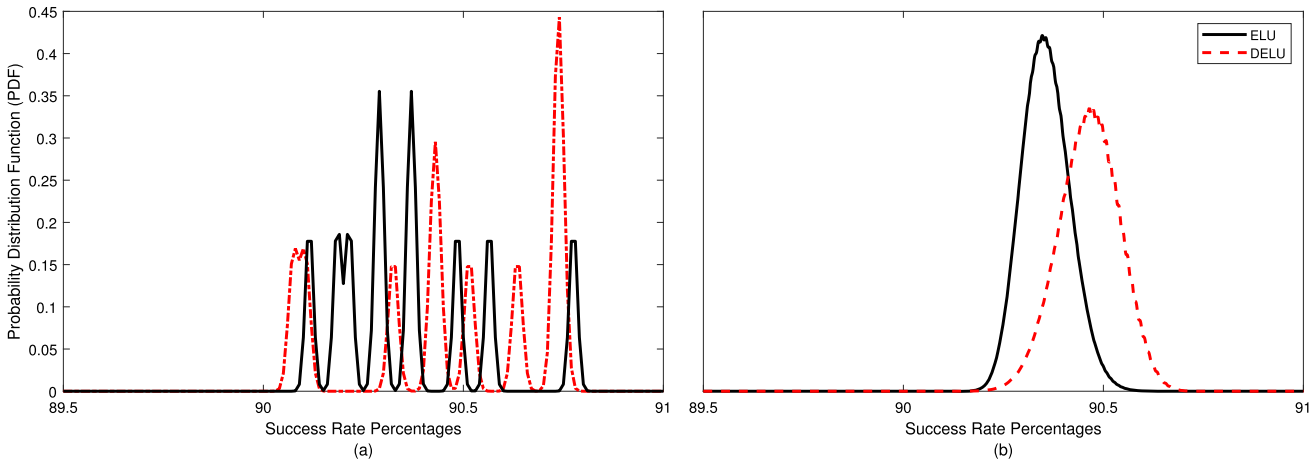

Fig. 11 **a** Contains two different Gaussian-kernel probability distribution functions for ELU and DELU. **b** The result of 9 convolution operations for these two PDFs with themselves, for both ELU and DELU separately. It is obvious that the PDF evolves to Gaussian-like shape even with convolving only 10 PDFs, as expected by CLT

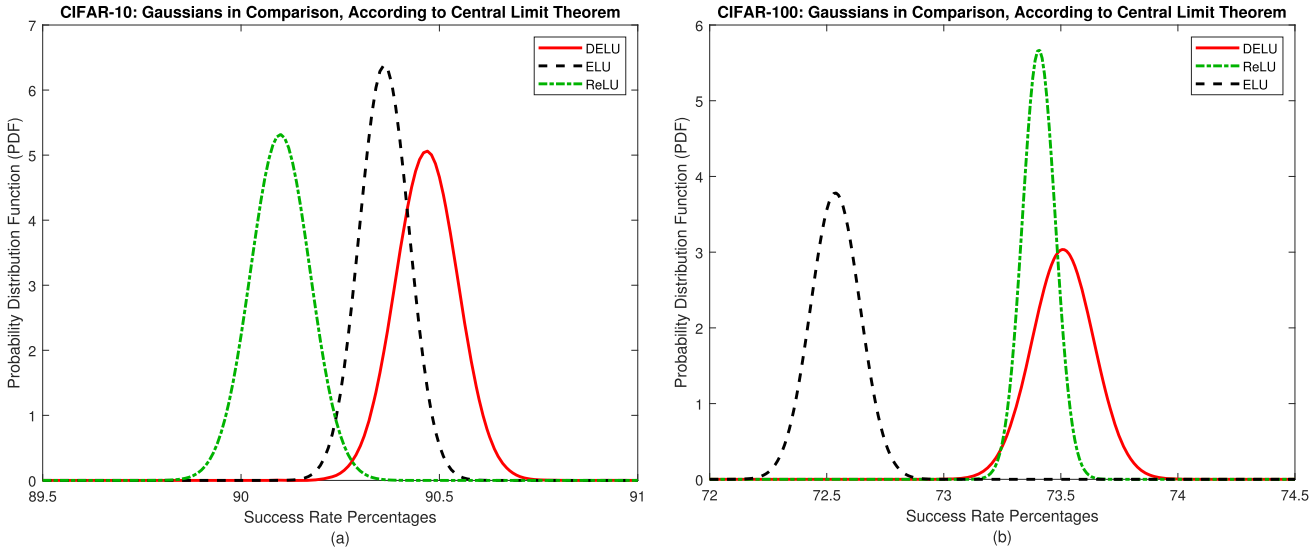

Fig. 12 Gaussians related to ReLU, ELU and DELU based on Table 1 at left (**a**) and Table 2 at right (**b**). Figure 12a and Fig. 11b are evidently similar, as CLT expected

We first note that in the following analysis, we utilize the results given in Table 1 or 2. Secondly, we assume that the Gaussian distributions given in Fig. 12 are representative of the corresponding experiments given in Tables 1 and 2. To determine which hypothesis is more likely for the experiments given in Table 1 or 2, we utilize a common hypothesis testing methodology which is frequently considered in Detection and Estimation Theory [40].

Making a mathematical claim about two different sets is not easy as a comparison between two scalar numbers. There can be outliers which may not represent the actual result, especially when a maximum or minimum operator is used to compare them. To model sets as random variables and claim results, especially when means are close, we need to introduce Z-score analysis to make a good hypothesis testing and reach a mathematical resolution. To summarize, Z-score can be named as a scalar that defines an area under a Gaussian PDF, which corresponds to probability value [40]. Definition of Z-score with multiple Gaussian curves in comparison is given later in this part in (20) with details.

Z-score analysis is used in different sources as transformations [41] and in the field of detection and estimation [42]. When it is possible to model the sets as Gaussian random variables, using Z-score analysis on their PDFs may help to decide between sets. To give an example, making a decision about which of two different Gaussian PDFs produce higher values can be a simple case of application. In our work, we construct our hypotheses to decide which activation function is better, defined by

giving higher success rates. By using obtained success rates and their standard deviation values, PDFs are constructed and employed for Z-score analysis. In this way, our assertions will be more precise than reaching a decision by only using presented success tables.

First, let us define "Percent of Confidence," which is abbreviated to PoC for convenience. PoC is the strength of hypothesis H1 in comparison with H0. Hence, if for example PoC is 90%, we can infer that H1 holds with 90% likelihood as compared to H0. To assess PoC, it is common in hypothesis testing applications to define a complementary variable; α, also called as Alpha, is given below:

$$\alpha = 1 - PoC. \tag{19}$$

If there is an underlying Gaussian PDF related to the hypothesis given above, α can be related to another quantity called Z-score, and it is given below:

$$\text{Z-score} = \frac{\text{MEAN}_{\text{DELU}} - \text{MEAN}_{\text{ReLU}}}{(\text{STD}_{\text{ReLU}})/\sqrt{\text{Sample amount}}}. \tag{20}$$

Formula for calculating the Z-score (20) is given in [43]. Note that (20) is utilized for the comparison of DELU and ReLU activation functions. Likewise for the comparison of DELU and ELU, the same formula will be utilized after replacing $\text{MEAN}_{\text{ReLU}}$ and STD_{ReLU} with MEAN_{ELU} and STD_{ELU}, respectively. In this formula $\text{MEAN}_{\text{DELU}}$ is the mean value of success rates for DELU activation function-applied networks, likewise $\text{MEAN}_{\text{ReLU}}$ (similarly STD_{ELU}) is the value for ReLU (similarly ELU). STD_{ReLU} (similarly $\text{MEAN}_{\text{ReLU}}$) stands for the standard deviation of success rates obtained for ReLU (similarly ELU) activation function applied networks. *Sample amount* is the number of trained networks, which is 10 for classification networks.

The relation between Z-score and Alpha score, evaluated by upper tail test using Gaussian distributions, can be given as in Table 9, for details see [43].

Now let us consider the experiments given in Table 1. For a comparison between ELU and DELU, let us compute the Z-score given by (20) for this pair. Noting that $\text{STD}_{\text{ELU}} = 0.1976$, $\text{MEAN}_{\text{DELU}} = 90.468$ and $\text{MEAN}_{\text{ELU}} = 90.361$ and Sample amount $= 10$, we compute a Z-Score $= 3.162$, which corresponds to $\alpha = 0.001$ from Table 9. Now, from (19) we can compute the value of PoC. We can interpret this result as follows: PoC for H1 hypothesis, which is replacing ELU with DELU does increase success rates, is 99.9%.

Table 9 Alpha vs. Z-score, for hypothesis tests

	Upper-tail test						
Alpha	0.1	0.05	0.025	0.01	0.005	0.001	0.0001
Z-score	1.282	1.645	1.96	2.326	2.576	3.09	3.719

A follow-up comparison can be done with ReLU and DELU for this case. Noting that $\text{STD}_{\text{ReLU}} = 0.2372$, $\text{MEAN}_{\text{DELU}} = 90.468$, $\text{MEAN}_{\text{ReLU}} = 90.099$ and Sample amount $= 10$, we compute a Z-Score $= 4.919$ which is higher than previous, 3.162. This result, as expected from graphs in Fig. 12, verifies that DELU does increase success rates, this time with 99.99% PoC in comparison with ReLU. Picking the minimum, we can say that DELU has 99.9% PoC for increasing the success rate, in comparison with ReLU or ELU.

A similar comparison between ReLU and DELU for the CIFAR-100 results given in Table 2 yields Z-Score $= 1.465$, from values $\text{STD}_{\text{ReLU}} = 0.2223$, $\text{MEAN}_{\text{DELU}} = 73.509$, $\text{MEAN}_{\text{ReLU}} = 73.406$ and Sample amount $= 10$. It corresponds to $\alpha = 0.1$ from Table 9. This can be interpreted as follows: PoC for H1 hypothesis (replacing ReLU with DELU increases success rates) is 90%.

Another follow-up comparison can be done with ELU and DELU for that case too. Note that $\text{STD}_{\text{ELU}} = 0.3335$, $\text{MEAN}_{\text{DELU}} = 73.509$, $\text{MEAN}_{\text{ELU}} = 72.537$ and Sample amount $= 10$, we compute a Z-Score $= 9.2166$ which is much higher than previous one, 1.465. This result, as expected from graphs in Fig. 12, verifies that DELU does increase success rates, this time with 99.99% PoC in comparison with ELU. Picking the minimum, we can say DELU has 90% PoC for increasing the success rate, compared to ReLU or ELU.

So long as only one-tail tests would be less strong, DELU must be treated with a similar understanding of hypothesis testing, where replacing DELU is considered as the statistically significant change or not. New hypotheses and correspondent negative Z-scores given below reflect the complementing nature.

- H0: Replacing ReLU (or ELU) with DELU does not increase success rates
- H1: Replacing ReLU (or ELU) with DELU does increase success rates

$$Neg.Z\text{-Score} = \frac{\text{MEAN}_{\text{ReLU}} - \text{MEAN}_{\text{DELU}}}{(\text{STD}_{\text{DELU}})/\sqrt{\text{Sample amount}}} \tag{21}$$

Similarly, Z-score formula changes for these hypotheses as above. Note that (21) is used for comparison of DELU and ReLU. Likewise, for comparison with ELU, same formula will be used after replacing $\text{MEAN}_{\text{ReLU}}$ with MEAN_{ELU}.

The relation between Neg.Z-score and Alpha score, evaluated by upper tail test using Gaussian distributions, can be given as in Table 10, for details see [43]. This

Table 10 Alpha vs. Z-score, for hypothesis tests

	Lower-tail test						
Alpha	0.1	0.05	0.025	0.01	0.005	0.001	0.0001
Z-score	− 1.282	− 1.645	− 1.96	− 2.326	− 2.576	− 3.09	− 3.719

table will be used for adding 'other tail comparison' into one-tail comparisons. In this way we will have two-tail comparisons for CIFAR networks.

Now let us consider the experiments given in Table 1 again. For a comparison between ELU and DELU, let us compute the Z-score given by (21) for this pair. Note that $\text{STD}_{\text{DELU}} = 0.2492$, $\text{MEAN}_{\text{DELU}} = 90.468$, $\text{MEAN}_{\text{ELU}} = 90.361$ and Sample amount $= 10$, we compute a Z-Score $= -1.358$, which corresponds to $\alpha = 0.1$ from Table 10. As before, by using (21) we can compute the PoC. We may interpret this result as follows: PoC for H0 hypothesis, which is replacing ELU with DELU does not increase success rates, is 90%.

A follow-up comparison can be made with ReLU and DELU for this case too. Noting that $\text{STD}_{\text{DELU}} = 0.2492$, $\text{MEAN}_{\text{DELU}} = 90.468$, $\text{MEAN}_{\text{ReLU}} = 90.099$ and Sample amount $= 10$, we compute a Z-Score $= -4.683$ which is magnitude-wise higher than previous one, -1.358. This verifies that DELU does increase success rates, this time with 99.99% PoC in comparison with ReLU. Picking the minimum, we can say that DELU has 90% PoC for increasing the success rate, in comparison with ReLU or ELU for two-tail comparison.

Applying this logic for the CIFAR-100 results obtained in Table 2 as done before gives us Z-Score $= -0.784$, with values $\text{STD}_{\text{DELU}} = 0.4157$, $\text{MEAN}_{\text{DELU}} = 73.509$, $\text{MEAN}_{\text{ReLU}} = 73.406$ and Sample amount $= 10$. This corresponds to $\alpha = 0.25$ and thus, at least 75% as PoC for H0 hypothesis, which is replacing ReLU with DELU, does not increase success rates. Therefore, the lowest of two values is picked and we can finally, statistically infer that DELU became more successful than ReLU with a PoC value of 75% at least, for the experiments conducted with CIFAR-10 and CIFAR-100 datasets.

Follow-up comparison for that case is comparing ELU and DELU for Table 2 results, which yields Z-Score $= -7.394$, with values $\text{STD}_{\text{DELU}} = 0.4157$, $\text{MEAN}_{\text{DELU}} = 73.509$, $\text{MEAN}_{\text{ELU}} = 72.537$ and Sample amount $= 10$.It corresponds to $\alpha = 0.0001$ from Table 10, sets PoC as 99.999% and picking the minimum value means that DELU is more successful than ReLU or ELU with a PoC value of 75% at least, for experiments with CIFAR datasets.

In addition to the success tables, these statistical analyses make it clear that DELU is a promising activation function and a candidate for common use, although it has slightly slower application time in training. Still, it yields higher performance while keeping its training and test timings close to others.

5.3 Optimizing the parameters of DELU

Standardized DELU uses fixed α and β, respectively, 1 and 2, in its final design. In this part, we explored different DELU alternatives. In the first stage, we varied both values within chosen limits. z_c is determined automatically according to choice of α and β which are from intervals $\alpha \in [0.5, 2]$ and $\beta \in [1, 3]$, with $\beta > \alpha$ condition. We varied α and β by incrementing them by 0.5 for both values within these intervals. By this framework, we obtained Fashion MNIST and CIFAR-10 mean success rates for each DELU alternative with (α, β) pairs, with 10 trained networks which use these alternatives.

For having $z_c > 0$, we must have the relation $\beta > \alpha$ for every positive (α, β) pair. With this condition, we chose 11 different (α, β) pairs. In the end of our search, pairs with $\alpha = 0.5$ and $\alpha = 2$ have performed less successful than $\alpha = 1$ and $\alpha = 1.5$, regardless of the β value chosen. So, we have continued with the second stage of our DELU optimization with $\alpha = 1$ and $\alpha = 1.5$.

Table 11 Fashion MNIST and CIFAR-10 mean success rates of different DELU alternatives

	F.MNIST Results			CIFAR-10 Results	
	$\alpha = 1$	$\alpha = 1.5$		$\alpha = 1$	$\alpha = 1.5$
$\beta = 1.75$	93.745%	93.688%	$\beta = 1.75$	89.725%	89.696%
$\beta = 1.875$	93.824%	93.763%	$\beta = 1.875$	89.796%	89.548%
$\beta = 2$	93.784%	93.810%	$\beta = 2$	90.468%	89.881%
$\beta = 2.125$	93.855%	93.857%	$\beta = 2.125$	89.832%	89.861%
$\beta = 2.25$	93.807%	93.764%	$\beta = 2.25$	89.781%	89.844%
$\beta = 2.375$	93.866%	93.715%	$\beta = 2.375$	89.726%	89.899%
$\beta = 2.5$	93.849%	93.878%	$\beta = 2.5$	89.824%	90.030%
$\beta = 2.625$	93.776%	93.746%	$\beta = 2.625$	89.723%	89.966%
$\beta = 2.75$	93.911%	93.760%	$\beta = 2.75$	89.689%	89.897%
$\beta = 2.875$	93.801%	93.875%	$\beta = 2.875$	89.656%	89.650%
$\beta = 3$	93.868%	93.922%	$\beta = 3$	89.670%	90.048%
$\beta = 3.125$	93.840%	93.802%	$\beta = 3.125$	89.653%	89.868%
$\beta = 3.25$	93.928%	93.806%	$\beta = 3.25$	89.575%	89.760%
$\beta = 4$	93.763%	93.812%	$\beta = 4$	89.486%	89.740%

Similar to the approach followed in the design of MPELU activation function [19], we came up with different (α, β) pairs and selected a good couple among the alternatives in the first stage. Still, it is possible to go deeper with a finer search. For this purpose, we present Table 11 in the second stage. As seen in this table, limiting α values to two best ones and varying β value in a finer search gave in a better result than original DELU activation function for the Fashion MNIST dataset; with (α, β) = (1, 3.25) pair, 93.928% and 89.575% mean successes are obtained in Fashion MNIST and CIFAR-10, respectively. However, original parameter set of (α, β) = (1, 2) is still better than the others for the CIFAR-10 dataset, by a margin around 0.4%. In the result, we came up with a method to search (α, β) values to find DELU alternatives and ended up with informative results on Fashion MNIST and CIFAR-10 in two stages. In this way, different choices of α and β values are also justified. Indeed, it is possible to go even deeper for the search of (α, β) with other methods, but it is left as a possible future work.

6 Conclusion

In this work we proposed a novel activation function: DELU, which can be considered as a generalization of classical activation functions ReLU and ELU.

To motivate the proposed activation function, we first give some heuristic and intuitive reasons which possibly may result in relatively higher success rates when utilized in a given neural network structure, as compared to its counterparts. In comparison with ReLU and ELU, usage of DELU increased the mean success rate 0.075–0.146% for Fashion MNIST dataset results, while the increase was 0.107% and 0.103% for CIFAR-10 and CIFAR-100 datasets, respectively. As results given in Tables 2, 3, 4 and 5 illustrate, DELU also gives the maximum success rate among those results obtained for aforementioned three activation functions in comparison with higher margins. The minimum success rates obtained for these experiments also pretty close to each other as seen in Tables 1, 2, 3, 4, 5 and 7. To sum up, image segmentation and classification examples performed in this section have suggested a performance increase by replacing ReLU or ELU with proposed activation function, DELU.

We performed these simulations with 6 activation functions and the proposed activation function DELU, and the results of these simulations are discussed in detail. CIFAR-10, CIFAR-100 and Fashion MNIST datasets used for training the network are introduced, network specialties for ResNet architectures are explained, and training conditions for these settings are given exclusively. Alongside those, comparison of network results is also presented for all seven activation functions exclusively. In addition to standard presentation of obtained test results, our work contains a statistical significance test based on CIFAR dataset results to compare ReLU, ELU and DELU activation function. With one-tail tests on CIFAR datasets, DELU activation function is better than ReLU and ELU with more than 90% percentage of confidence. In addition to this, with two-tail tests, it is still possible to confidently show that DELU performs better with at least 75% confidence rate using hypothesis testing methods on all CIFAR datasets used in this work. Also, ResNet networks are tried on Fashion MNIST dataset with same activation function set, and results are given for verification purposes in addition to CIFAR results. Also, we made an image segmentation example with Unet-like networks to separate cats and dogs from the background in Oxford IIIT-Pets Dataset, with all activation functions are again compared by their minimum, mean and maximum success rates out of 20 independent runs for 300 epochs training configuration. Here, DELU scored the highest mean success rate. On the mean success rates obtained, the highest is DELU with 61.299%, while ReLU follows it with a 61.251%, and MPELU is third with 60.968% in this experiment. These results indicate that, although we cannot claim with certainty that the usage of DELU or ReLU yields better results for the image segmentation problem, success rates are comparable with the results obtained for the structure considered in our experiments. It may be possible that for different architectures, using DELU may yield better or worse performance. However, this point needs to be carefully investigated beyond the scope of this article, but at least, it can be claimed that DELU cannot be declared worse than ReLU with these results.

The main contributions of our work can be listed as follows. First, we introduced a new activation function, DELU, in detail. Second, DELU is compared with different activation functions for various datasets, using well-known ResNet architectures for certain classification tasks. Third, statistical tests are used to show DELU's success in comparison with others by hypothesis testing for CIFAR networks. Fourth, we made an image segmentation task with DELU as well. Fifth, we made other analyses, including testing initialization insensitivity of DELU and exploring DELU alternatives. The results of these tests indicate that the utilization of the DELU may improve the success rates as compared to its counterparts from a statistical and empirical point of view.

In the near future, our goal will be determining the relation between different initialization techniques and activation functions, while expanding our research to various application areas and using different network types, such as recurrent neural network (RNN). This will help us to evaluate the efficiency of DELU activation function with

various initialization methods in different neural network architectures. In addition, a bounded version of DELU might be applicable to RNN units, such as LSTM cells, for different tasks like time-series prediction and neural language processing, as a possible future work. Key research directions are applying DELU into different network configurations and obtaining results, trying it out with state-of-the-art architectures and determining DELU's effect on them, continuing to optimize DELU with different setups and possibly, adaptively changing α and β values by varying them during the training process for possible improvement in different applications.

Acknowledgements This work was supported by the Scientific and Technological Research Council of Türkiye (TÜBİTAK). The authors appreciate the financial support of the TÜBİTAK. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the P6000 Quadro GPU used for this research.

Data availability Datasets used during this study are open to access in publicly available sources, which are given in the reference list as [25, 26, 32].

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Ding B, Qian H, Zhou J (2018) Activation functions and their characteristics in deep neural networks. Chin Control Decis Conf 2018:1836–1841. https://doi.org/10.1109/CCDC.2018.8407425
2. Alhassan AM, Zainon WMNW (2021) Brain tumor classification in magnetic resonance image using hard swish-based RELU activation function-convolutional neural network. Neural Comput Appl 33:9075–9087. https://doi.org/10.1007/s00521-020-05671-3
3. Çatalbaş B (2022) Control and system identification of legged locomotion with recurrent neural networks (Doctoral Dissertation). Retrieved from http://repository.bilkent.edu.tr/handle/11693/90921
4. Haykin S (1999) Neural networks: a comprehensive foundation. Prentice Hall, New Jersey, NY
5. Dubey SR, Singh SK, Chaudhuri BB (2022) Activation functions in deep learning: a comprehensive survey and benchmark. Neurocomputing 503:92–108. https://doi.org/10.1016/j.neucom.2022.06.111
6. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 807–814
7. Williams A (2017) The art of building neural networks. TheNewStack. https://thenewstack.io/art-building-neural-networks/
8. Zheng Q, Yang M, Yang J, Zhang Q, Zhang X (2018) Improvement of generalization ability of deep CNN via implicit regularization in two-stage training process. IEEE Access 6:15844–15869. https://doi.org/10.1109/ACCESS.2018.2810849
9. Li H, Zeng N, Wu P, Clawson K (2022) Cov-Net: a computer aided diagnosis method for recognizing COVID-19 from chest X-ray images via machine vision. Expert Syst with Appl 207:118029. https://doi.org/10.1016/j.eswa.2022.118029
10. Zhang K, Yang X, Zang J, Li Z (2021) FeLU: a fractional exponential linear unit. In: 2021 33rd Chinese Control and Decision Conference (CCDC), pp 3812–3817. https://doi.org/10.1109/CCDC52312.2021.9601925
11. Apicella A, Donnarumma F, Isgrò F, Prevete R (2021) A survey on modern trainable activation functions. Neural Netw 138:14–32. https://doi.org/10.1016/j.neunet.2021.01.026
12. Clevert D-A, Unterthiner T, Hochreiter S (2015) Fast and accurate deep learning by exponential linear units (ELUs). In: The International Conference on Learning Representations (ICLR), pp 1–14. https://doi.org/10.48550/arXiv.1511.07289
13. Qiumei Z, Dan T, Fenghua W (2019) Improved convolutional neural network based on fast exponentially linear unit activation function. IEEE Access 7:151359–151367. https://doi.org/10.1109/ACCESS.2019.2948112
14. Adem K (2021) P + FELU: flexible and trainable fast exponential linear unit for deep learning architectures. Neural Comput Appl 34:21729–21740. https://doi.org/10.1007/s00521-022-07625-3
15. Sakketou F, Ampazis N (2019) On the invariance of the SELU activation function on algorithm and hyperparameter selection in neural network recommenders. In: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, Cham, pp 673–685. https://doi.org/10.1007/978-3-030-19823-7_56
16. Ramachandran P, Zoph B, Le QV (2017) Searching for activation functions. arXiv preprint arXiv:1710.05941v2. https://doi.org/10.48550/arXiv.1710.05941
17. Zhou Y, Li D, Hou D, Kung SY (2021) Shape autotuning activation function. Expert Syst with Appl 171:114534. https://doi.org/10.1016/j.eswa.2020.114534
18. Alkhouly AA, Mohammed A, Hefny HA (2021) Improving the performance of deep neural networks using two proposed activation functions. IEEE Access 9:82249–82271. https://doi.org/10.1109/ACCESS.2021.3085855
19. Li K, Fan C, Li Y, Wu Q, Ming Y (2018) Improving deep neural network with multiple parametric exponential linear units. Neurocomputing 301:11–24. https://doi.org/10.1016/j.neucom.2018.01.084
20. Github (2018) Code for improving deep neural network with multiple parametric exponential linear units. Github. Retrieved from https://github.com/Coldmooon/Code-for-MPELU
21. Lu L, Shin Y, Su Y, Karniadakis G (2020) Dying ReLU and initialization: theory and numerical examples. arXiv preprint arXiv:1903.06733v3. https://doi.org/10.48550/arXiv.1903.06733
22. Billingsley P (1995) Probability and measure, 3rd edn. John Wiley & Sons, New York, NY
23. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep networks training by reducing internal covariate shift. In: International Conference on Machine Learning, pp 448–456, PMLR. https://doi.org/10.48550/arXiv.1502.03167
24. Alcaide E (2018) E-swish: adjusting activations to different network depths. arXiv: 1801.07145. https://doi.org/10.48550/arXiv.1801.07145
25. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Retrieved from www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf
26. Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747. https://doi.org/10.48550/arXiv.1708.07747

27. Shan S, Willson E, Wang B, Li B, Zheng B, Zhao BY (2019) Gotta catch 'em all: using concealed trapdoor to detect adversarial attacks on neural networks. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp 1–14
28. Ruiz P (2018) Understanding and visualizing ResNets. Towards Data Science. Retrieved from https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8
29. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. In: European conference on computer vision, Springer, Cham, pp 630–645. https://doi.org/10.1007/978-3-319-46493-0_38
30. Keras (n.d.) Trains a ResNet on the CIFAR10 dataset. Keras. Retrieved from https://keras.io/zh/examples/cifar10_resnet/
31. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp 1026–1034. https://doi.org/10.1109/ICCV.2015.123
32. Parkhi O, Vedaldi A, Zisserman A, Jawahar CV (2012) Cats and dogs. In: IEEE Conference on Computer Vision and Pattern Recognition (2th ed.). Retrieved from www.robots.ox.ac.uk/~vgg/data/pets/. https://doi.org/10.1109/CVPR.2012.6248092
33. Chollet F (2020) Image segmentation with a U-Net-like architecture. Keras. Retrieved from keras.io/examples/vision/oxford_pets_image_segmentation
34. He Y, Zhang X, Sun J (2017) Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE international conference on computer vision. https://doi.org/10.48550/arXiv.1707.06168
35. Salakhutdinov R, Larochelle H (2010) Efficient learning of deep boltzmann machines. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics
36. Papoulis A (1985) Probability, random variables, and stochastic processes, 2nd edn. McGraw-Hill, New York, NY
37. Epanechnikov VA (1969) Non-parametric estimation of a multivariate probability density. Theory Probab Appl 14(1):153–158
38. The Math Works Inc. (2021) Kernel Distribution. MathWorks. https://www.mathworks.com/help/stats/kernel-distribution.html
39. Poor HV (2013) An introduction to signal detection and estimation. Springer Science & Business Media, Berlin
40. DeVore GR (2017) Computing the Z score and centiles for cross-sectional analysis: a practical approach. J Ultrasound Med 36(3):459–473
41. Urolagin S, Sharma N, Datta TK (2021) A combined architecture of multivariate LSTM with Mahalanobis and Z-Score transformations for oil price forecasting. Energy 231:120963. https://doi.org/10.1016/j.energy.2021.120963
42. Adler K, Gaggero G, Maimaitijiang Y (2010) Distinguishability in EIT using a hypothesis-testing model. J Phys: Conf Ser 224(1):12056. https://doi.org/10.1088/1742-6596/224/1/012056
43. LaMorte WW (2017) Hypothesis testing: upper-, lower, and two tailed tests. Boston University School of Public Health. Retrieved from sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_hypothesistest-means-proportions/bs704_hypothesistest-means-proportions3.html