`Serf`: Towards better training of deep neural networks using log-Softplus ERror activation Function

Sayan Nag*
University of Toronto
nagsayan112358@gmail.com

Mayukh Bhattacharyya*
Stony Brook University
mayukh.bhattacharyya@stonybrook.edu

Anuraag Mukherjee*
IISER Mohali
anuraag.m107@gmail.com

Rohit Kundu*
UCR Riverside
rohit.kundu@email.ucr.edu

Abstract

Activation functions play a pivotal role in determining the training dynamics and neural network performance. The widely adopted activation function ReLU despite being simple and effective has few disadvantages including the Dying ReLU problem. In order to tackle such problems, we propose a novel activation function called `Serf` which is self-regularized and non-monotonic in nature. Like Mish, `Serf` also belongs to the Swish family of functions. Based on several experiments on computer vision (image classification and object detection) and natural language processing (machine translation, sentiment classification and multi-modal entailment) tasks with different state-of-the-art architectures, it is observed that `Serf` vastly outperforms ReLU (baseline) and other activation functions including both Swish and Mish, with a markedly bigger margin on deeper architectures. Ablation studies further demonstrate that `Serf` based architectures perform better than those of Swish and Mish in varying scenarios, validating the effectiveness and compatibility of `Serf` with varying depth, complexity, optimizers, learning rates, batch sizes, initializers and dropout rates. Finally, we investigate the mathematical relation between Swish and `Serf`, thereby showing the impact of pre-conditioner function ingrained in the first derivative of `Serf` which provides a regularization effect making gradients smoother and optimization faster.

1. Introduction

Activation functions are point-wise functions which play a crucial role in introducing non-linearity in neural networks. In a neural network, linear transformed inputs are passed through an activation function giving rise to non-linear counterparts. These non-linear point-wise activation functions are hugely responsible for the performance of neural networks. Thus, choosing a suitable activation function for better training and improved efficiency has always been an interesting area of research. Activation functions like $tanh$ and $sigmoid$ were widely used in previous works [26, 27, 33, 19]. However, they had some disadvantages including upper-boundedness. This paved the way for the development of an activation function widely known as Rectified Linear Unit (ReLU) [35]. Being simple yet effective, ReLU was not only easier to optimize compared to its contemporaries (sigmoid and tanh), but also showed better generalization and improved convergence properties, which led to its wide adoption.

ReLU however has few disadvantages including the infamous *Dying ReLU* phenomenon [29, 32]. The absence of any negative portion resulted in such a problem which can be noticed through a gradient information loss caused by collapsing the negative inputs to zero. On the other hand, ReLU is also non-differentiable which can result in inconsistencies during gradient-based optimization. Keeping those in mind, researchers have propounded various activation functions including leaky ReLU [32], PReLU[12], ELU [4], GELU [15], SELU [22], Swish [37], Mish [34], etc. Out of the aforementioned activation functions, Mish mostly outperforms its contemporaries including Swish. Mish has a continuous profile which renders better information propapagation as compared to ReLU. It was inspired by the self-gating property of Swish. As opposed to Swish, Mish possess a preconditioner which results in smoother gradients and better optimization.

*Denotes equal contribution

In this work, we have proposed a novel activation function called *Serf* which is non-monotonic and is also inspired by the self-gating property of Swish. We define Serf as $f(x) = x\,\mathrm{erf}(\ln(1+e^x))$ where erf is the error function [1]. Swish, Mish and Serf belong to the same family of activation functions possessing self-gating property. Like Mish, Serf also possess a pre-conditioner which results in better optimization and thus enhanced performance. Our experiments demonstrate that our proposed activation function Serf outperforms ReLU, Swish and even Mish for different standard architectures in a variety of tasks including image classification, object detection, graph node classification, machine translation, sentiment classification and multi-modal entailment, involving varied datasets. We have also conducted ablation studies on MNIST [25] and CIFAR-10 datasets [24] to demonstrate the efficiency of Serf over Swish and Mish.

2. Related Work

One of the mostly used activation functions is Rectified Linear Units (ReLU) [35]. Originally proposed for Restricted Boltzmann Machines, this activation function gained prominence because of its simplicity and effectiveness and eventually replaced the sigmoid and tanh units. Despite being computationally efficient, it is not entirely devoid of shortcomings. In order to address those, Leaky ReLU (LReLU) was introduced which replaced the constant zero portion of the ReLU function with a linear function thereby 'leaking' some information [32]. LReLU showed superior performance compared to ReLU, and the performance was further enhanced when the slope of the negative part was learnt as an extra parameter using Parametric ReLU (PReLU) [12]. However, lower boundedness is important in order to render strong regularization effects which was absent in both LReLU and PReLU. Furthermore, similar to ReLU, they are also not differentiable.

Keeping these aspects in mind, researchers proposed activation functions like Exponetial Linear Units (ELU) [4] and Scaled Exponential Linear Units (SELU) [22]. ELU and SELU possess better convergence characteristics along with a saturation plateau in its negative region. However, these activation functions have found to be incompatible with Batch Normalization (BN) [18].

Finally, using self-gating property Swish was proposed which addressed the aforementioned drawbacks to a greater extent abreast demonstrating superior results compared to the previous established activation functions [37]. Belonging to the same class as Swish, another activation function called Mish was proposed which performed equally well or better than Swish in most of the computer vision tasks [34]. Our proposed activation function, Serf, is also inspired from the self-gating mechanism and thus belongs to the Swish-like class of functions. It has been shown experimentally that our proposed Serf outperforms other activation functions in a variety of computer vision and natural language processing tasks.

3. Serf

3.1. Motivation

Activation functions introduce non-linearity in the neural networks and they play a very important role in the overall performance of a network. ReLU has been the most widely used activation function in neural networks. However, it suffers from several disadvantages, the most noticeable one being the *dying ReLU* phenomenon. This problem ensued from the missing negative part in the ReLU activation function which restrains the negative values to zero. At the same time, ReLU is *not* continuously differentiable. Furthermore, ReLU is a non-negative function. This creates a *non-zero mean* problem where the mean activation larger than zero. Such an issue is not desirable for network convergence [4].

In order to address these aforesaid problems to some extent, in the recent past several new activation functions emerged including leaky ReLU, ELU, Swish, etc. Swish is seemingly an ideal candidate for an activation function with properties including non-monotonicity and ability to preserve small negative weights abreast maintaining a smooth profile. Similar to Swish, activation functions like GELU [15] has gained popularity especially in the transformer based architectures used both in the fields of Computer Vision (ViT [7] and MLP Mixer [43]), as well as Natural Language Processing (GPT-2 [36] and GPT-3 [3]). Another activation function which rose to prominence due to its performance in state-of-the-art classification and object detection tasks, is Mish. Mish has its roots in Swish and was developed by methodical analysis over the attributes that led to the efficacy of Swish.

Taking inspiration from the development of Mish, we propose an activation function called Serf. Serf is defined as:

$$f(x) = x\,\mathrm{erf}(\ln(1+e^x)) \tag{1}$$

3.2. Properties

Serf is bounded below and unbounded above. Serf is smooth, non-monotonic and differentiable. It also preserves small portion of negative weights. Serf is inspired by Swish and Mish where the *self-gating* property has been used to multiply the output of a non-linear function of an input with the same non-modulated input. Self-gating is advantageous because it requires only a single-scalar input, whilst normal gating requires multiple two-scalar inputs [37].

- **Upper unboundedness**: Activation functions like $tanh$ and $sigmoid$ have upper bounds. So, initializa-

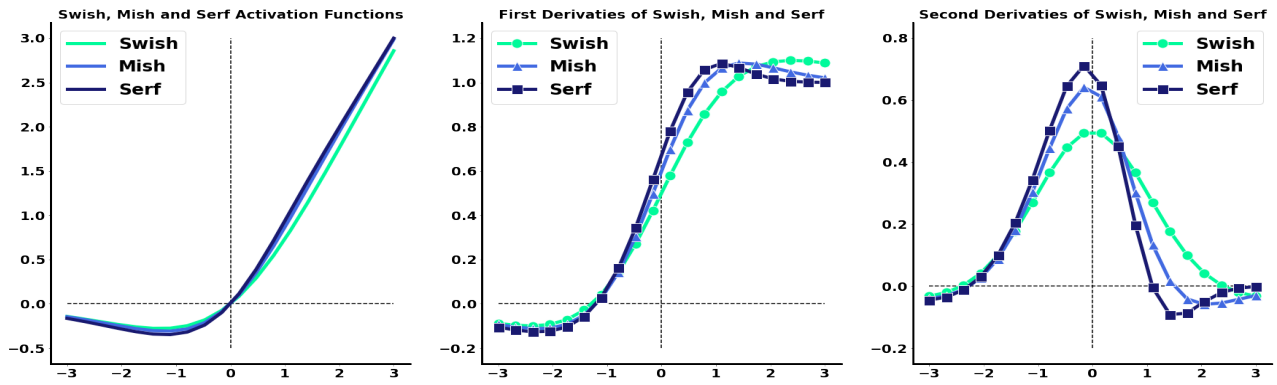

Figure 1. Activation functions (Left), first derivatives (Middle) and second derivatives (Right) for Swish, Mish and `Serf`.

tion should happen in the linear regime of these activation functions. Such a property is not desirable since it leads to saturation while training due to near-zero gradients [10]. ReLU being unbounded above attempted to avoid the saturation problem. This is a crucial attribute which can be noticed in all the successors of the ReLU function like leakyReLU, GELU, Swish, Mish, etc. `Serf` also possesses this feature, with its positive side as an approximate linear function of the input (see Figure 1). This makes `Serf` a good candidate for an activation function.

- **Lower boundedness**: Activation functions must posses lower bounds in order to provide strong regularization effects. However, in the ReLU activation function a neuron receiving negative input will always output zero eventually become *dead* or inactive and hence useless. This is referred to as the *dying ReLU* phenomenon [29, 32]. It usually happens when the learning rate is high or if there is a large negative bias. By preserving of a small portion of negative information, `Serf` mitigates the aforementioned problem furthermore resulting in better expressivity and improved gradient flow. The negative bound for `Serf` is approximately 0.3484 (see Figure 1).

- **Differentiability**: Unlike ReLU, `Serf` is continuously differentiable. This is beneficial owing to the fact that it avoids singularities and any concomitant ill-effects during gradient-based optimization.

- **Preconditioner**: `Serf` is closely related to Swish which can be noticed in its first derivative. The first derivative of `Serf` is given as:

$$f'(x) = \frac{2}{\sqrt{\pi}} e^{-\ln((1+e^x))^2} x\sigma(x) + \frac{f(x)}{x} = p(x)\,\text{swish}(x) + \frac{f(x)}{x} \quad (2)$$

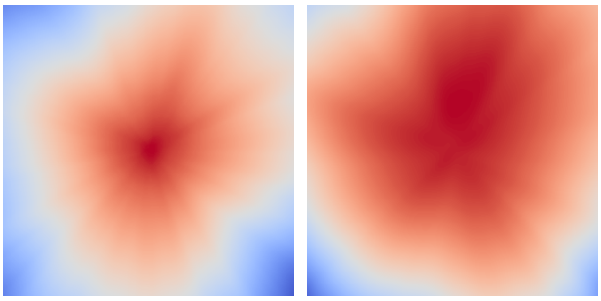

Figure 2. Output landscapes of a randomly initialized 6-layered neural network with ReLU (Left) and `Serf` (Right) activations.

Here, σ is the sigmoid function and $p(x)$ is a preconditioner function. Preconditoners make the gradients smoother and have been previously used extensively in optimization problems. The inverse of a symmetric positive definite matrix has been used as a preconditioner in case of gradient descent. Application of such preconditioners makes the objective function smoother thereby increasing the rate of convergence [2]. Therefore, the strong regularization effect contributed by such preconditoner in case of `Serf` makes the gradients smoother and optimization faster, thereby outperforming Swish as can be noticed in the experiments.

Mish also has a precondtioner which makes it perform better than Swish. The difference between Mish and Swish is that in `Serf` we used the error function (erf) whereas in Mish $tanh$ function is used. `Serf`, however, outperforms Mish in most experiments (see Experiments). We speculate that `Serf`'s preconditioner function renders better regularization effects than that of Mish.

- **Smoothness**: Smooth loss landscapes indicates easier optimization with less local optima and hence better generalization minimizing influence of initializations and learning rates. The output landscapes of a ran-

domly chosen 6-layered neural network with ReLU and Serf activation functions has been shown in Figure 2. It is to be noted that output landscape is indicative of the loss landscape. We randomly initialize a 6-layered neural network, where we pass the x and y coordinates of each point in a grid as input, and plot the scalar network output for each grid point. For ReLU activation function, the output landscape of the neural network has sharp transitions in contrast to that of Serf. This conforms to the enhanced performance of Serf as compared to ReLU.

Please refer to Section 5 for more in depth analysis.

4. Experiments

In this section we will demonstrate and compare the performance of our proposed activation function, Serf when used in different state-of-the-art architectures on image, sequence and graph datasets for disparate tasks. All the scores reported are averages of 3 distinct runs which ensures robustness of the results. The experiments were carried out on a NVIDIA Tesla V100 with 32GB RAM.

We evaluate Serf on multiple tasks ranging from traditional image classification to Machine Translation. We also perform a set of ablations to gauge the performances of the activation functions on the basis of different configurations of hyperparameters.

4.1. Image Classification

For image classification, we have considered different standard architectures applied on CIFAR-10, CIFAR-100 and ImageNet. The experiments have been separated according to type of architectures and datasets.

CIFAR-10/100: We have considered different deep learning architectures (for CIFAR-10: SqueezeNet [17], Resnet-50 [13], WideResnet-50-2 [46], ShuffleNet-v2 [30], ResNeXt-50 [45], Inception-v3 [41], DenseNet-121 [16], MobileNet-v2 [40] and EfficientNet-B0 [42]; for CIFAR-100: Resnet-164 [14], WideResnet-28-10 [46], DenseNet-40-12 [16], Inception-v3 [41]) with three disparate activation functions, namely, ReLU (baseline), Mish and Serf (proposed). This has been done for image classification task on CIFAR-10 and CIFAR-100 datasets where for each network we have only changed the activation functions and have kept every other parameter constant for fair comparisons. Tables 1 and 2 show that Serf consistently outperformed both ReLU and Mish activation functions across all the architectures used in the experiment for both CIFAR-10 and CIFAR-100 datasets.

We have also used two recent architectures namely MLP Mixer [43] and Compact Convolutional Transformers (CCT) [11] and evaluated the performance on these with a shorter training schedule. We have trained and evaluated the MLP Mixer on CIFAR-10 with two different activation functions, GELU (standard to MLP Mixer) and Serf. The Top-1 % and Top-5 % Accuracy values (see Table 3) suggest that Serf's performance is comparable to GELU's (baseline) performance for MLP Mixer. We have also trained and evaluated CCT on CIFAR-10 with three different activation functions - ReLU, Mish and Serf. Serf clearly outperforms ReLU (baseline) and Mish in this case (see Table 3). The results indicate that Serf is a better activation function for transformer based architectures.

Methods	ReLU	Mish	**Serf (Ours)**
SqueezeNet	84.14	85.98	**86.32**
Resnet-50	86.54	87.03	**88.07**
WideResnet-50-2	86.39	86.57	**86.73**
ShuffleNet-v2	83.93	84.07	**84.55**
ResNeXt-50 (32 × 4d)	87.25	87.97	**88.49**
Inception-v3	90.93	91.55	**92.89**
DenseNet-121	88.59	89.05	**89.07**
MobileNet-v2	85.74	86.39	**86.61**
EfficientNet-B0 (Swish)	78.26	78.02	**78.41**

Table 1. Top-1 % Accuracy values of different state-of-the-art methods for different activation functions on CIFAR-10

Methods	ReLU	Mish	**Serf (Ours)**
Resnet-164	74.55	75.02	**75.13**
WideResnet-28-10	76.32	77.03	**77.54**
DenseNet-40-12	73.68	73.91	**74.16**
Inception-v3	71.54	72.38	**72.95**

Table 2. Top-1 % Accuracy values of different state-of-the-art methods for different activation functions on CIFAR-100

Model	Activations	Top-1 % Acc	Top-5 % Acc
MLP-Mixer	GELU	64.14	**96.71**
	Serf	**64.36**	96.59
CCT	ReLU	79.05	97.72
	Mish	80.02	**98.70**
	Serf	**80.23**	98.65

Table 3. Top-1 and Top-5 % Accuracy values (classification) after 10 epochs of training MLP-Mixer for GELU (SOTA) and Serf activation functions on CIFAR-10 test dataset and 50 epochs of training Compact Convolutional Transformer (CCT) for ReLU, Mish and Serf functions on CIFAR-10 test dataset.

ImageNet: State-of-the-art architectures for ImageNet[5] classification utilize ReLU [35] activation function. For comparison purposes, we have selected 3 widely used ar-

Model	Activations	Top-1% Acc	Top-5% Acc
Resnet-50	ReLU	74.16	90.28
	Serf (Ours)	**75.34**	**91.71**
ResNeXt-50	ReLU	75.84	92.32
	Serf (Ours)	**76.81**	**93.19**
EfficientNet-B0	Swish	75.42	91.55
	Serf (Ours)	**75.60**	**91.83**

Table 4. Top-1% and -5% Accuracy values on the ImageNet dataset.

chitectures, namely, Resnet-50 [13], ResNeXt-50 [45], and finally Efficient-Net B0 [42]. However, it is to be noted that baseline activation function for Efficient-B0 is Swish [37]. Results in Table 4 demonstrate that our activation function outperforms the baseline in all the 3 cases suggesting that Serf works well even with large datasets like ImageNet.

4.2. Object Detection

Object detection is considered one of the important visual scene understanding tasks. In our case, we have considered Pascal VOC and MS-COCO datasets for object detection task using YOLOv3 [38] and tiny YOLOv3 architectures. We have evaluated Serf against Leaky ReLU which is intrinsic to the YOLOv3 framework. For fair comparisons we have only changed the activation function keeping other hyperparameters fixed as outlined in [38]. Mean Average Precision (MAP) scores in Table 5 clearly indicate that our proposed Serf outperforms the baseline leaky ReLU based architectures in the object detection task for both Pascal VOC and MS-COCO datasets.

Dataset	Model	Activations	MAP@.5	MAP@.5:.95
VOC	YOLOv3	LeakyReLU	74.0	47.3
		Serf (Ours)	**76.6**	**50.1**
	YOLOv3 Tiny	LeakyReLU	50.3	21.9
		Serf (Ours)	**51.4**	**22.7**
COCO	YOLOv3	LeakyReLU	51.2	32.5
		Serf (Ours)	**53.3**	**33.4**
	YOLOv3 Tiny	LeakyReLU	32.7	15.2
		Serf (Ours)	**33.5**	**15.5**

Table 5. Mean Average Precision scores for different object detection models on the Pascal VOC and MS-COCO datasets. LeakyReLU is instrinsic to YOLO framework.

4.3. Semi-supervised Node Classification

Following the implementations outline in [21], we have considered 3 different datasets, namely CITESEER, CORA and PUBMED for semi-supervised node classification using three disparate activation functions, namely, ReLU (baseline), Mish and Serf (proposed). All training parameters and hyper-parameters have been kept same as mentioned in [21] for fair comparisons. Table. 6 shows that Serf either performed equally well or better than both ReLU and Mish activation functions across the three different datasets, thereby indicating versatility of the proposed activation function.

Dataset	ReLU	Mish	**Serf (Ours)**
CORA	81.5	**81.7**	**81.7**
CITESEER	70.3	71.3	**71.7**
PUBMED	79.0	79.3	**79.4**

Table 6. Top-1 % Accuracy values of different state-of-the-art GNN semi-supervised node classification methods for different activation functions on CORA, CITESEER and PUBMED.

4.4. Machine Translation, Sentiment Classification & Multi-modal Entailment

In this section, we have demonstrated the effectiveness of our proposed Serf activation function in Machine Translation and Sentiment Classification tasks. We have considered 3 different architectures and datasets.

For Machine Translation we have used a sequence-to-sequence Transformer [44] Encoder-Decoder based model trained (for 20 epochs) on the the Multi30k dataset for German-English translation [9]. For comparison purposes, we have considered ReLU, GELU, Mish and Serf (proposed) and observed that Serf outperformed the remaining three activation functions as suggested by the BLEU scores shown in the Table 7.

For sentiment classification, we have considered two datasets, namely imdb movie review sentiment and Pol Emo 2.0 sentiment datasets. For the imdb movie review sentiment dataset [31], we have considered: (i) a simple architecture consisting of a 1D conv net with a text embedding layer which we have trained using three different activation functions and noticed that Serf outperformed the other two activation functions (ReLU and Mish) suggesting that Serf also works well with simple architectures (see Table 8), and (ii) a 4-layer Transformer model which we have also trained for 20 epochs for each of the three activation functions eventually obtaining the best results for the proposed Serf function (see Table 8). For the Pol Emo 2.0 sentiment database [23], we have used a BERT based model [6] with a classification head for two different activation functions, Mish and Serf. The Precision, Recall and F1-scores suggest that Serf performed equally well or better than Mish for this task (see Table 9).

For multi-modal entailment task, we have used the multi-modal entailment database, recently introduced by

the Google Research [1]. We have used a smaller variant of the original BERT model. The code used for this purpose is available at [2]. We have used two activation functions for comparison purposes: GELU and `Serf`. Table 10 shows the accuracies on test dataset averaged over 5 runs (each trained for 10 epochs). The accuracy values suggest that `Serf` performs marginally better than GELU in this case.

Score	ReLU	GELU	Mish	**`Serf` (Ours)**
BLEU	35.55	35.62	35.36	**36.06**

Table 7. BLEU scores of seq2seq Transformer model (after training for 20 epochs) for different activation functions on Multi30k test dataset.

Model	ReLU	Mish	**`Serf` (Ours)**
1D conv with text-embedding	85.36	85.99	**86.18**
4-layer Transformer model	88.82	88.99	**89.03**

Table 8. Top-1 % Accuracy values of 1D conv net with a text embedding layer and 4-layer Transformer model for ReLU, Mish and `Serf` on imdb movie review sentiment dataset.

Activation	Precision	Recall	F1-score
Mish	0.8374	0.8329	**0.8346**
`Serf` (Ours)	**0.8377**	**0.8330**	0.8342

Table 9. Precision, Recall and F1-scores for different activation functions for sentiment classification using BERT on Pol Emo 2.0 sentiment database.

Metric	GELU	**`Serf` (Ours)**
Mean Accuracy	85.28	**85.42**

Table 10. Mean Accuracy values of GELU and `Serf` based architectures for multi-modal entailment task.

4.5. Ablations

Model hyperparameters play an important role in the training and optimization of neural networks thus having direct consequences in the generalizability of a network. Such hyper-parameters include network depths, network widths, type of weight initializations, dropout rates, batch sizes, learning rates, and optimizers. Here we analyze and compare the impacts of different hyper-parameters on our chosen networks with three different activation functions namely Swish, Mish, and `Serf`. We have used MNIST and CIFAR-10 datasets for this purpose.

[1]https://github.com/google-research-datasets/recognizing-multimodal-entailment

[2]https://github.com/sayakpaul/Multimodal-Entailment-Baseline

4.5.1 MNIST

- **Dense Units**: The number of dense units refers to the number of neurons present in a dense layer. In this case we have used a 4 layered architecture with one dense layer followed by a Batch Normalization Layer and SGD [39] as an optimizer. We observe that as the number of dense units increases, the model complexity increases and `Serf` outperforms Swish and Mish (Fig 3). This suggests that `Serf` works well with complex models. This has also been noticed in other experiments.

- **Dropout Rates**: As the dropout rate increases, the overall performance for all three activation functions drop, however, the performance degradation for `Serf` is relatively less than Swish and Mish (Fig 3).

- **Initializers**: The performance of `Serf` is better than both Swish and Mish in all except for random uniform initialization (Fig 3). This suggests that `Serf` is a better candidate compared to its contemporaries.

- **Learning Rates**: With varying learning rates, `Serf` performs better than both Swish and Mish (Fig 3). Particularly, with higher learning rates, the degradation is quite pronounced in Swish and not that much in Mish and `Serf`. We have used SGD [39] as an optimizer in this case.

- **Optimizers**: In this case, with varying optimizers, the overall performance of `Serf` is equal or marginally better than Swish and Mish (Fig 3). Performance drop can be noticed for all three activation functions in case of Adagrad optimizer [8].

- **Number of layers**: In this case, each dense layer was followed by a Batch Normalization layer. As the number of dense layers increases, models become complex and optimization becomes difficult. The degradation in the performances for all the three different activation functions conforms the aforementioned fact. However, `Serf` maintained a significantly higher accuracy as compared to Swish and Mish (Fig 3). [3].

4.5.2 CIFAR-10

We have used a ResNet-18 model with a dense layer and a classification head in tandem. The results are obtained with training the model over multiple runs for 20 epochs, which gives a decent convergence point.

- **Batch Size**: We observed that with decreasing training batch sizes, the performance for all the competing activation functions drop (Fig 4), however, `Serf` holds

[3]Code: https://anonymous.4open.science/r/`Serf`-3630/

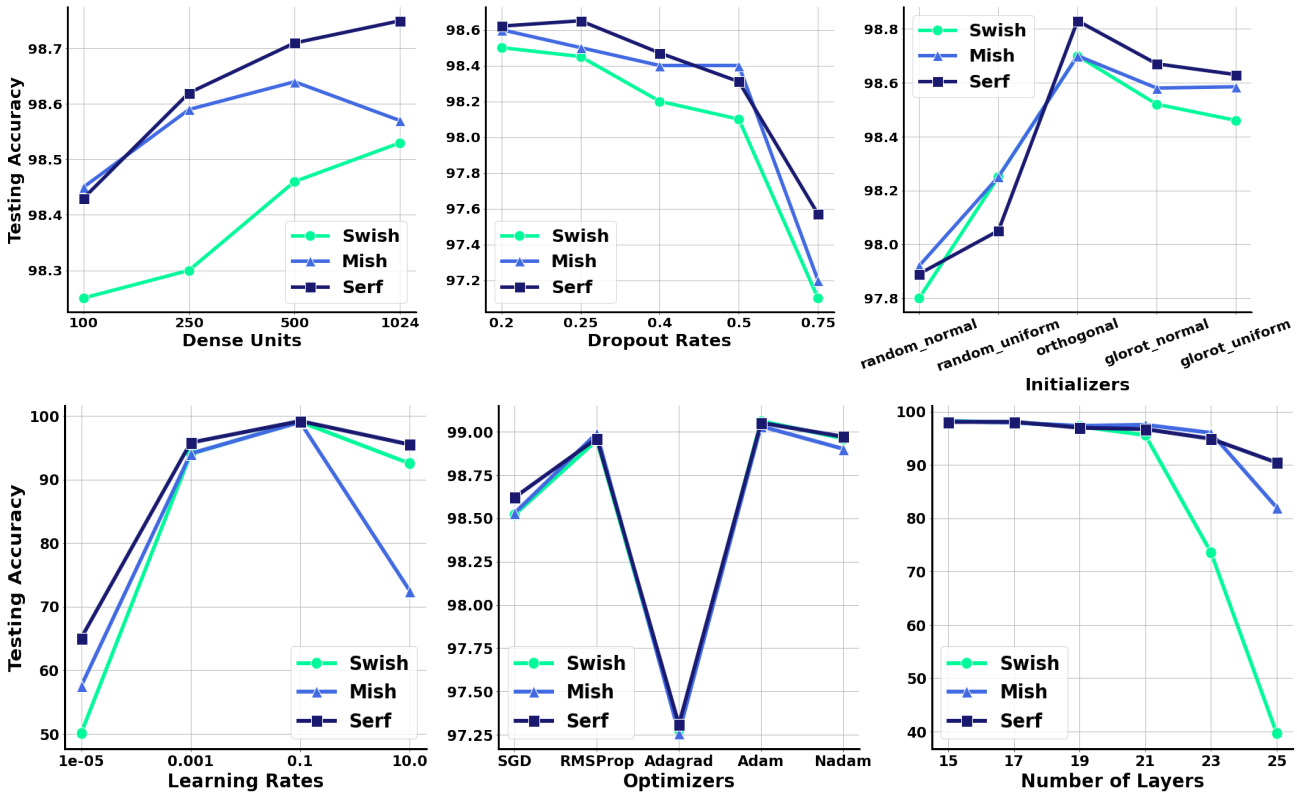

Figure 3. Ablations for MNIST dataset. Top: Testing Accuracies vs Dense Units (Left), Dropout Rates (Middle) and Initializers (Right) for Swish, Mish and `Serf`. Bottom: Testing Accuracies vs Learning Rates (Left), Optimizers (Middle) and Number of Layers (Right) for Swish, Mish and `Serf`.

the better position out of all three over all the batch sizes. Adam [20] is used as the optimzer here.

- **Optimizers**: In this case, with varying optimizers, the overall performance of `Serf` is equal or marginally better than Swish and Mish (Fig 4). Performance drop can be noticed for all three activation functions in case of [8] and SGD optimizers [39].
- **Learning Rates**: `Serf` is observed to be performing better or equal to both Mish and Swish on all of the learning rates evaluated barring 0.1 where a steeper drop is observed in `Serf` compared to the other two activation functions (Fig 4). Adam [20] is used as the optimzer here.

5. Analysis

Typically first derivatives of activation function are responsible for parameter updates, for instance, in a stochastic gradient-based optimization scheme. The value of a gradient plays an important role in this regard. Final update of a parameter is dependent on this value multiplied with a learning rate which determines the speed of convergence. Furthermore, in deep neural networks Batch Normalization (BN) is used in conjunction with activation functions. Considering the best practices, that is, normalized input, BN attempts to maintain the mean output close to 0 and the output standard deviation close to 1. This ensures that almost around 87% of the values are within 1.5 standard deviation of the mean. Considering this range in the Fig 1, we can observe that the first derivative for Serf lies above Mish which lies above Swish. Therefore, we can consider that the first derivative of Serf is $\mu(> 1)$ times more than that of Mish. Therefore, the effective learning rate is increased for Serf which leads to faster and better convergence.

Moreover, the pre-conditioner equation from the Equation (2) can be re-written as:

$$g(x) = \frac{2}{\sqrt{\pi}} e^{-(ln(1+e^x))^2} \sigma(x) \frac{e^{2(ln(1+e^x))} + 1}{e^{2(ln(1+e^x))} - 1} \quad (3)$$

$$f'(x) = g(x) Mish(x) + \frac{f(x)}{x} \quad (4)$$

where $g(x)$ acts as a pre-conditioner. Mish has been shown to have a pre-conditioning effect over Swish which actually helped in the performance. Here, similarly we show that Serf has a preconditioning effect over Mish making the

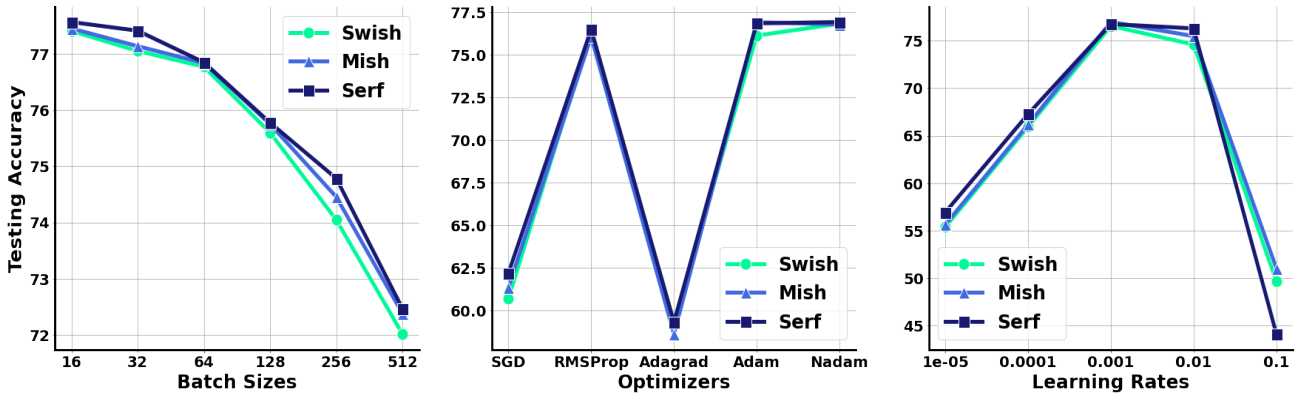

Figure 4. Ablations for CIFAR-10 dataset. Testing Accuracies vs Batch Sizes (Left), Optimizers (Middle) and Learning Rates (Right) for Swish, Mish and `Serf`.

gradient even smoother and thereby providing stronger regularization effect.

The same $p(x)$ (pre-conditioner of Serf over Swish) from the Equation (2) can also be re-written as:

$$p(x) = r(x)\Delta(x) \tag{5}$$

$$\begin{aligned} r(x) &= \frac{e^{(-ln(1+e^x))^2}\left(e^{2ln(1+e^x)} + e^{-2ln(1+e^x)} + 2\right)}{2\sqrt{\pi}} \\ &= \frac{e^{(ln(1+e^x))^2}}{2\sqrt{\pi}}\left((1+e^x)^2 + (1+e^x)^{-2} + 2\right) \\ &\geq \frac{e^{(ln(1+e^x))^2}}{2\sqrt{\pi}}(2+2) \quad \text{(Young's inequality)} \\ &\geq \frac{4}{2\sqrt{\pi}} \quad \text{(since } e^{(ln(1+e^x))^2} \geq 1\text{)} \\ &> 1 \end{aligned} \tag{6}$$

where $\Delta(x)$ is the pre-conditioner of Mish over Swish and $r(x) \geq 1$ (in the operating range) which indicates $p(x)$ is better.

6. Computational Time Efficiency

We performed computational efficiency study for the prominent activation functions used in the community. Results presented in Table 11 show that despite its superior performance across a variety of tasks, the computational time of `Serf` is more as compared to ReLU, whereas, it is comparable with GELU, Mish and Swish. We have used system with a NVIDIA Tesla V100 GPU and 32 GB RAM to perform these experiments.

7. Conclusions and Future Works

In this paper, we proposed a novel activation function which has demonstrated properties such as upper unboundedness, lower boundedness, non-monotonicity and smoothness which are the desired for an activation function. Experimental results with different state-of-the-art architectures on varied datasets for multiple disparate tasks demonstrate that the proposed `Serf` outperforms the baseline ReLU performance and as well as other activation functions like Swish, Mish and GELU by a good margin. The results can be improved with desired hyperparameters for `Serf` which can be obtained with a hyperparameter search.

Activation	Forward Pass	Backward Pass
ReLU	5.41 ± 0.42 µs	5.72 ± 1.59 µs
LeakyReLU	5.63 ± 0.74 µs	6.08 ± 0.89 µs
GELU	8.91 ± 1.22 µs	9.31 ± 1.56 µs
Swish	8.52 ± 1.37 µs	10.85 ± 2.23 µs
Mish	7.59 ± 2.66 µs	9.03 ± 2.92 µs
`Serf`(Ours)	7.62 ± 1.54 µs	10.91 ± 1.68 µs

Table 11. Computational time comparison for different activation functions for a 32 × 32 × 3 input in a PreActResnet-18 model.

`Serf` opens up a plethora of opportunities to move forward. Probable future works include (1) the understanding of the importance and contribution of pre-conditioner as a regularizer and how modifying it can have an impact on the final results; this can lead to the development of more effective activation functions, (2) the development and exploration of probabilistic version of `Serf` as shown in [28], (3) the development of parameterized `Serf` like PReLU, and finally (4) comparison the performance of `Serf` with other contemporary activation functions for tasks such as image super-resolution, image reconstruction, etc. Overall, `Serf` is a simple, effective and versatile activation function which can be incorporated in any neural network for better training and performance gains.

References

[1] Larry C Andrews. *Special functions of mathematics for engineers*, volume 49. Spie Press, 1998.

[2] Owe Axelsson and Gunhild Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48(5):499–523, 1986.

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[8] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[9] Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74. Association for Computational Linguistics, 2016.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[11] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[17] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[19] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[22] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Proceedings of the 31st international conference on neural information processing systems*, pages 972–981, 2017.

[23] Jan Kocoń, Monika Zaśko-Zielińska, and Piotr Miłkowski. Polemo 2.0 sentiment analysis dataset for conll. 2019.

[24] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[25] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[26] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Joonho Lee, Kumar Shridhar, Hideaki Hayashi, Brian Kenji Iwana, Seokjun Kang, and Seiichi Uchida. Probact: A probabilistic activation function for deep neural networks. *arXiv preprint arXiv:1905.10761*, 5:13, 2019.

[29] Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

[30] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[31] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[32] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[33] José Mira and Francisco Sandoval. *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks, Malaga-Torremolinos, Spain, June 7-9, 1995: Proceedings*, volume 930. Springer Science & Business Media, 1995.

[34] Diganta Misra. Mish: A self regularized non-monotonic neural activation function. *arXiv preprint arXiv:1908.08681*, 4:2, 2019.

[35] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[36] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[37] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[38] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[39] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[40] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[41] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[42] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[43] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[45] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[46] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.