TanhSoft—Dynamic Trainable Activation Functions for Faster Learning and Better Performance

KOUSHIK BISWAS¹, SANDEEP KUMAR¹,2, SHILPAK BANERJEE³, AND ASHISH KUMAR PANDEY³

¹Department of Computer Science, IIIT Delhi, New Delhi 110020, India
²Shaheed Bhagat Singh College, University of Delhi, New Delhi 110021, India
³Department of Mathematics, IIIT Delhi, New Delhi 110020, India
Corresponding author: Koushik Biswas (koushikb@iiitd.ac.in)

ABSTRACT
Deep learning, at its core, contains functions that are the composition of a linear transformation with a nonlinear function known as the activation function. In the past few years, there is an increasing interest in the construction of novel activation functions resulting in better learning. In this work, we propose three novel activation functions with learnable parameters, namely TanhSoft-1, TanhSoft-2, and TanhSoft-3, which are shown to outperform several well-known activation functions. For instance, replacing ReLU with TanhSoft-1, TanhSoft-2, and TanhSoft-3 improves top-1 classification accuracy by 6.06%, 5.75%, and 5.38% respectively on VGG-16(with batch-normalization), by 3.02%, 3.25% and 2.93% respectively on PreActResNet-34 in CIFAR-100 dataset, by 1.76%, 1.93%, and 1.82% respectively on WideResNet 28-10 in Tiny ImageNet dataset. TanhSoft-1, TanhSoft-2, and TanhSoft-3 outperformed ReLU on mean average precision (mAP) by 0.7%, 0.8%, and 0.6% respectively in object detection problem on SSD 300 model in Pascal VOC dataset.

INDEX TERMS
Deep learning, neural network, activation function.

I. INTRODUCTION
Artificial neural networks (ANNs) have occupied the center stage in deep learning in the recent past. ANNs are made up of several hidden layers, and each hidden layer consists of several neurons. At each neuron, an affine linear map is composed with a nonlinear function known as activation function. During the training of an ANN, the linear map is optimized; however, an activation function is usually fixed in the beginning, along with the architecture of the ANN. There has been an increasing interest in developing a methodical understanding of activation functions, particularly with regards to the construction of novel activation functions and identifying mathematical properties leading to better learning [1].

An activation function is considered good if it can generalise better on a variety of datasets, ensure faster convergence and improve neural network performance, which leads to more accurate results. At the early stage of deep learning research, researchers used shallow networks (fewer hidden layers), along with tanh or sigmoid were used as activation functions. As the research progressed and deeper networks (multiple hidden layers) came into fashion to achieve challenging tasks, the Rectified Linear Unit (ReLU) [2], [3] emerged as the most popular activation function. Despite its simplicity, deep neural networks with ReLU have learned many complex and highly nonlinear functions with high accuracy.

To overcome the shortcomings of ReLU (non-zero mean, negative missing, unbounded output, to name a few [4]), and to increase the accuracy considerably in a variety of tasks in comparison to networks with ReLU, many new activation functions have been proposed over the years. Many of these new activation functions are variants of ReLU, for example, Leaky ReLU [5], Exponential Linear Unit (ELU) [6], Parametric Rectified Linear Unit (PReLU) [7], Randomized Leaky Rectified Linear Units (RReLU) [8] and Inverse Square Root Linear Units (ISRLU) [9], Flexible ReLU (FReLU) [10]. In the recent past, some activation functions constructed from tanh or sigmoid have
A. MOTIVATION

Swish, GELU, and Mish are a few recently proposed activations, which gained popularity in the deep learning community. They share similar mathematical properties like smoothness, non-linearity, non-monotonic, small and bounded negative output. GELU is a popular activation widely used in Natural language processing tasks and recently used in GPT-2 [12] architecture for text generation. Swish was found by a group of researchers from Google by automated neural architecture search and shown promising results compared to ReLU. Mish is recently proposed by Misra, which shown some promising results on computer vision problems, especially on object detection task in YOLO v4 [13] model. Recently, Homomorphic encryption friendly a new novel non-linear activation function [14] is proposed by Obla et al. based on Softplus [15] polynomial approximation. Motivated by these activation functions, we are interested in constructing some activations which share similar properties while having better performance in a wide range of deep learning problems (like Image classification, Object Detection, Semantic Segmentation, Machine Translation etc.) on different datasets and models when compared to widely used activations like ReLU, Swish, GELU, and Mish. We start with a few functions like \( xtanh(be^{cx}) \), \( tanh(ax)ln(1 + e^x) \), \( ln(1 + e^{tanh(cx)}) \), \( tanh(ax + be^{cx})ln(1 + e^x) \), \( tanh(be^{cx})ln(1 + e^x) \) etc (most of these functions can be generated from the generalized functional form \( tanh(ax + be^{cx})ln(d + e^x) \), which we have constructed as a function generator) and conducted experiments on MNIST [16], CIFAR10 [17] and CIFAR100 [17] database on ResNet-34 [18], VGG-16 [19], DenseNet [20], MobileNet V2 [21], and Inception V3 [22] models. We found that \( xtanh(be^{cx}) \), \( tanh(ax)ln(1 + e^x) \), and \( ln(1 + e^{tanh(cx)}) \) performed remarkably well when compared to ReLU and Swish while the other functions, we extracted from the generator, either fails to perform better than ReLU or performs similar to ReLU. So we decided to further investigate these three functional forms as activations and run experiments on several standard benchmarking datasets as well as other public datasets on different deep learning problems (more detailed experiments with these three activations are given in the experiment section).

II. RELATED WORKS

An activation function that can improve neural network models performance is an active field of research. It is always hard to find the best activation function. In earlier days, Tanh and Sigmoid were mostly used as activations in networks. ReLU [2] was first proposed by Nair and Hinton in 2010, and since then, ReLU is the widely used activation function in neural network models due to its simplicity. ReLU produces a positive outcome for positive inputs while zero for negative inputs, and due to this, ReLU undergoes from vanishing gradient problem, which is known as dying ReLU [5]. Several activation functions have been suggested by researchers to overcome this problem. Leaky ReLU [5] has been proposed with a small negative linear function for negative input, and it shows promising results compared to ReLU. PReLU [7] has been introduced with a modification of Leaky ReLU and added a trainable linear part for negative inputs. Later,

| Activation Function | Zero-Centered | Non-Monotonic | Non-zero Negative | In negative axis, small and bounded | Continuous | Smooth | Trainable |
|---------------------|--------------|--------------|------------------|-------------------------------------|-----------|-------|----------|
| ReLU                | Yes          | No           | No               | Zero outcome and bounded             | Yes       | No    | No       |
| Swish               | Yes          | Yes          | Yes              | Yes                                 | Yes       | Yes   | Yes      |
| Leaky ReLU          | Yes          | No           | Yes              | Yes                                 | Yes       | Yes   | No       |
| ELU                 | Yes          | No           | Yes              | Yes                                 | Yes       | Yes   | No       |
| Softplus            | No           | No           | No               | Positive outcome and bounded         | Yes       | Yes   | No       |
| Mish                | Yes          | Yes          | Yes              | Yes                                 | Yes       | Yes   | No       |
| GELU                | Yes          | Yes          | Yes              | Yes                                 | Yes       | Yes   | No       |
| TanhSoft-1          | Yes          | Yes          | Yes              | Yes                                 | Yes       | Yes   | No       |
| TanhSoft-2          | Yes          | Yes          | Yes              | Yes                                 | Yes       | Yes   | No       |
| TanhSoft-3          | Yes          | Yes          | Yes              | Yes                                 | Yes       | Yes   | Yes      |

TABLE 1. The relationship and difference between the proposed activation’s and previously proposed widely used activation functions.
Some of the commonly used activation functions are ReLU [8], ISReLU [9], FReLU [10], PReLU [23], SiLU [24], ELU [6], and GELU [25]. These functions have been introduced recently, and they demonstrate their improved model performances. Mish [26] which has been introduced recently, has shown some improvement over ReLU and Swish [11].

There are several functions that have been proposed in recent years in order to better fit the problem at hand. Real-world datasets are noisy or challenging, and it always difficult to construct the best activation function to generalize on random datasets. It is hard to say whether an activation function will generalize successfully and replace ReLU on challenging or noisy datasets. Though there may be merit in having a custom activation function corresponding to the problem at hand, but yet it is beneficial to identify activation functions that generalize to several real-world data sets, making it easier to implement. Hence we concentrate on three activation functions, namely, TanhSoft-1, TanhSoft-2, and TanhSoft-3, and establish their generalizability and usefulness over other conventional activation functions. In what follows, we discuss the properties of these activations, experiments with complex models, and a comparison with a few other widely used activation functions.

### III. TANHSOFT-1, TANHSOFT-2, AND TANHSOFT-3 & THEIR PROPERTIES

The standard ANN training process involves tuning the weights in the linear part of the network; however, there is merit in the ability to custom design activation functions, to better fit the problem at hand. Real-world datasets are noisy or challenging, and it is always difficult to construct the best activation function to generalize on random datasets. It is hard to say whether an activation function will generalize successfully and replace ReLU on challenging or noisy datasets.

There have been a few trainable activations proposed like Adjustable Generalized Hyperbolic Tangent [27], Sigmoidal selector [28] etc. Later, Leaky ReLU, ELU, ReLU were modified by PReLU, PELU, and FReLU, respectively, by introducing trainable parameter(s). Recently, in 2017, Swish [11], a trainable activation was found using exhaustive search [29], and reinforcement learning techniques [30], like Adjustable Generalized Hyperbolic Tangent [27], and Swish [11].

Swish [11] was found using exhaustive search [29] and reinforcement learning techniques [30], and it has been introduced recently, has shown some improvement over ReLU and Swish. Mish [26] which has been introduced recently, has shown some improvement over ReLU and Swish.

Though there may be merit in having a custom activation function, and it is hard to say whether an activation function will generalize successfully and replace ReLU on challenging or noisy datasets. Though there may be merit in having a custom activation function corresponding to the problem at hand, but yet it is beneficial to identify activation functions that generalize to several real-world data sets, making it easier to implement.

Hence we concentrate on three activation functions, namely, TanhSoft-1, TanhSoft-2, and TanhSoft-3, and establish their generalizability and usefulness over other conventional activation functions. In what follows, we discuss the properties of these activations, experiments with complex models, and a comparison with a few other widely used activation functions.

TanhSoft-1, TanhSoft-2, and TanhSoft-3 are defined as

\[
\begin{align*}
\text{TanhSoft-1} & : \quad \mathcal{F}_1(x; \alpha) := \tanh(\alpha x) \text{Softplus}(x) \\
\text{TanhSoft-2} & : \quad \mathcal{F}_2(x; \beta, \gamma) := x \tanh(\beta e^{\gamma x}) \\
\text{TanhSoft-3} & : \quad \mathcal{F}_3(x; \delta) := (1 + e^{\delta x} \tanh(\delta x)) \tanh(\delta x)
\end{align*}
\]

The corresponding derivatives are

\[
\begin{align*}
\frac{d}{dx} \mathcal{F}_1(x; \alpha) & = \tanh(\alpha x) \frac{e^{\alpha x}}{1 + e^{\alpha x}} \\
\frac{d}{dx} \mathcal{F}_2(x; \beta, \gamma) & = \tanh(\beta e^{\gamma x}) \\
\frac{d}{dx} \mathcal{F}_3(x; \delta) & = \frac{e^{\delta x} \tanh(\delta x) + \delta e^{\delta x} \sech^2(\delta x)}{1 + e^{\delta x} \tanh(\delta x)}
\end{align*}
\]

Figures 1, 2 and 3 show the graph for \(\mathcal{F}_1(x; \alpha)\), \(\mathcal{F}_2(x; \beta, \gamma)\), and \(\mathcal{F}_3(x; \delta)\) activation functions for different values of \(\alpha\) and \(\beta\), \(\gamma\) and \(\delta\) respectively. Plots of the first derivative of \(\mathcal{F}_1(x; \alpha)\), \(\mathcal{F}_2(x; \beta, \gamma)\), and \(\mathcal{F}_3(x; \delta)\) are given in Figures 5, 6, and 7 for different values of \(\alpha\) and \(\beta\), \(\gamma\) and \(\delta\) respectively. A comparison between \(\mathcal{F}_1(x; \alpha)\), \(\mathcal{F}_2(x; \beta, \gamma)\), \(\mathcal{F}_3(x; \delta)\) and Swish and their first derivatives are given in Figures 4 and 8. The authors of [26] have reported unstable training behaviour for a specific function which can be obtained from TanhSoft-1, however, we tested and failed to find any such instability. Also, in [31] the authors have mentioned a special case which can be obtained from TanhSoft-2.

The authors have proposed several special cases which can be obtained from TanhSoft-2 and TanhSoft-3 plays a major role and controls the slope of the curve in both positive and negative axes as evident from Figures 1, 2 and 3. Like Swish, \(\mathcal{F}_1(x; \alpha)\), \(\mathcal{F}_2(x; \beta, \gamma)\), and \(\mathcal{F}_3(x; \delta)\) are both smooth, non-monotonic activation functions and bounded below.

\[
\lim_{\gamma \to \infty} \mathcal{F}_2(x; \beta, \gamma) = \text{ReLU}(x), \quad \forall x \in \mathbb{R}
\]

Also, The class of neural networks with TanhSoft-1 or TanhSoft-2 or TanhSoft-3 activation function is dense in \(\mathcal{C}(K)\), where \(K\) is a compact subset of \(\mathbb{R}^n\) and \(\mathcal{C}(K)\) is the space of all continuous functions over \(K\) (see [32]).

The proof follows from the following proposition as all three proposed activations are non-polynomial.

**Proposition (Theorem 1.1 in Kidger and Lyons, 2019 [33])** Let \(\rho : \mathbb{R} \to \mathbb{R}\) be any continuous function. Let \(N_h^\rho\) represent the class of neural networks with activation function \(\rho\), with \(n\) neurons in the input layer, one neuron in the output layer, and one hidden layer with an arbitrary number of neurons. Let \(K \subseteq \mathbb{R}^n\) be compact. Then \(N_h^\rho\) is dense in \(\mathcal{C}(K)\) if and only if \(\rho\) is non-polynomial.

### IV. EXPERIMENTS WITH TANHSOFT-1, TANHSOFT-2, AND TANHSOFT-3

In this work we have initialized and updated hyper-parameter values of TanhSoft-1, TanhSoft-2, and TanhSoft-3 using the backpropagation algorithm (see [7]). For a single layer, the gradient of a hyper-parameter \(\theta\) is:

\[
\frac{\partial E}{\partial \theta} = \sum_x \frac{\partial E}{\partial F(x)} \frac{\partial F(x)}{\partial \theta}
\]

where \(E\) is the objective function, \(\theta \in \{\alpha, \beta, \gamma, \delta\}\) and \(F(x) \in \{\mathcal{F}_1(x; \alpha), \mathcal{F}_2(x; \beta, \gamma), \mathcal{F}_3(x; \delta)\}\).

We have considered several models and datasets to measure the performance of \(\mathcal{F}_1(x; \alpha)\), \(\mathcal{F}_2(x; \beta, \gamma)\), and \(\mathcal{F}_3(x; \delta)\)
and have compared with seven baseline widely used activation functions. A brief description about baselines are as follows:

- **Rectified Linear Unit (ReLU):** The rectified linear unit (ReLU) activation function was first introduced by Nair and Hinton [2], Hahnloser et al. [3] and it is one of the widely used activation function. ReLU suffers from a vanishing gradient problem known as dying ReLU. ReLU is defined as

  \[ f(x) = \max(0, x). \]  

- **Leaky Rectified Linear Unit:** Leaky Rectified Linear Unit (Leaky ReLU) was proposed by Mass et al. [5]. Leaky ReLU has introduced an non-zero gradient in the negative axis to overcome the vanishing gradient and dead neuron problems of ReLU. LReLU is defined as

  \[ f(x) = \begin{cases} 
  x & x > 0 \\ 
  0.01x & x \leq 0. 
  \end{cases} \]  

- **Exponential Linear Units:** Exponential Linear Units (ELU) was proposed by Clevert et al. [6]. ELU is defined as

  \[ f(x) = \begin{cases} 
  x & x > 0 \\ 
  \alpha(e^x - 1) & x \leq 0. 
  \end{cases} \]  

  where \( \alpha \) is a hyper-parameter.

- **Swish:** Swish is a non-monotonic, smooth function introduced by Ramachandran et al. [11]. Swish is
defined as
\[ f(x) = x \sigma(\alpha x) = \frac{x}{1 + e^{-\alpha x}}. \] (12)
where \( \alpha \) is a trainable or constant hyper-parameter. If \( \alpha = 1 \), then Swish becomes SiLU \([24]\) activation.

- **Softplus**: Softplus was proposed by Zheng et al. \([15]\), Dugas et al. \([35]\) which is a smooth activation function and has non-zero gradient and defined as
\[ \text{Softplus}(x) = \log(1 + e^x). \] (13)

- **GELU**: GELU was introduced by Hendrycks and Gimpel \([25]\) and defined as
\[ \text{GELU}(x) = 0.5x(1 + \tanh(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3))). \] (14)

- **MISH**: Mish has been introduced recently by Mishra \([26]\) and defined as
\[ \text{Mish}(x) = x \tanh(\text{Softplus}(x)) = x \tanh(\log(1 + e^x)) \] (15)

In the following subsections, we have provided experimental results of TanhSoft-1, TanhSoft-2, and TanhSoft-3 with baseline activation functions such as ReLU, Leaky ReLU, ELU, Softplus, Swish, GELU, and Mish for different deep learning problems like Image classification, Object detection, Semantic segmentation, and Machine translation. We have initialized \( \alpha = 0.87 \) for TanhSoft-1, \( \beta = 0.75, \gamma = 0.75 \) for TanhSoft-2, and \( \delta = 0.85 \) for TanhSoft-3 (see \([7]\)) and updated these hyper-parameter values via backpropagation during training as mentioned in equation (8). In the following subsections, we will provide details of our experimental setup, framework, and results. All the experiments were conducted on an NVIDIA Tesla V-100 GPU with 16GB RAM.

### A. IMAGE CLASSIFICATION

We have reported results for image classification problem in some widely used standard datasets like MNIST \([16]\), Fashion MNIST \([36]\), SVHN \([37]\), CIFAR10 \([17]\), CIFAR100 \([17]\), and Tiny Imagenet \([38]\).

1) **MNIST**

The MNIST \([16]\) database contains image data of handwritten digits from 0 to 9. The dataset contains 60k training and 10k testing 28×28 grey-scale images. A 8-layer customised homogeneous convolutional neural network (CNN) architecture with 3×3 kernels for CNN layers and 2×2 kernels for pooling layers are being used. We have used channel depths of size 128 (twice), 64 (thrice), 32 (twice), a dense layer of size 128, Max-pooling layer(thrice), batch-normalization \([39]\), and dropout \([40]\) on the custom CNN architecture. Data augmentation method is not used. The results are reported in Table 2.

| Activation Function | 5-fold mean accuracy (%) on MNIST test data |
|---------------------|------------------------------------------|
| TanhSoft-1          | 99.40                                    |
| TanhSoft-2          | 99.34                                    |
| TanhSoft-3          | 99.37                                    |
| ReLU                | 99.14                                    |
| Swish               | 99.18                                    |
| Leaky ReLU(α = 0.01)| 99.20                                    |
| ELU                 | 99.10                                    |
| Softplus            | 99.05                                    |
| Mish                | 99.23                                    |
| GELU                | 99.21                                    |

2) **FASHION MNIST**

Fashion-MNIST \([36]\) is a popular computer vision database consisting of 28×28 pixels grey-scale images, consists of ten fashion items in each class.

| Activation Function | 5-fold mean accuracy (%) on Fashion MNIST test data |
|---------------------|------------------------------------------|
| TanhSoft-1          | 93.52                                    |
| TanhSoft-2          | 93.40                                    |
| TanhSoft-3          | 93.37                                    |
| ReLU                | 92.90                                    |
| Swish               | 92.92                                    |
| Leaky ReLU(α = 0.01)| 92.95                                    |
| ELU                 | 92.85                                    |
| Softplus            | 92.40                                    |
| Mish                | 93.17                                    |
| GELU                | 93.12                                    |

It has 60k training images and 10k testing images. Fashion-MNIST provides a more challenging classification problem than MNIST. The data augmentation method is not used. We have considered the same CNN architecture used in the MNIST dataset for this database as well for training and testing purpose and, the results are reported in Table 3.

3) **THE STREET VIEW HOUSE NUMBERS (SVHN) DATABASE**

SVHN \([37]\) is a popular image database consists of real-world house numbers of Google’s street view images with 32×32 RGB images. The database has 73257 training images and 26032 testing images. The database has a total of 10 classes. We have considered the same CNN architecture used in the MNIST dataset for this database as well for training and testing purpose and, the results are reported in Table 4.

| Activation Function | 5-fold mean accuracy (%) on SVHN test data |
|---------------------|------------------------------------------|
| TanhSoft-1          | 95.36                                    |
| TanhSoft-2          | 95.52                                    |
| TanhSoft-3          | 95.43                                    |
| ReLU                | 95.14                                    |
| Swish               | 95.23                                    |
| Leaky ReLU(α = 0.01)| 95.20                                    |
| ELU                 | 95.15                                    |
| Softplus            | 95.08                                    |
| Mish                | 95.33                                    |
| GELU                | 95.20                                    |
4) CIFAR

The CIFAR [17] is a popular computer vision dataset consists of $32 \times 32$ colored images, with total 60k images and divided into 50k training and 10k test images. There are two type of CIFAR dataset - CIFAR10 and CIFAR100. CIFAR10 dataset has 10 classes with 6000 images per class while CIFAR100 has 100 classes with 600 images per class. Top-1 accuracy for mean of 10 runs have been reported on CIFAR10 dataset in Table 5 & 6 and on CIFAR100 dataset in Table 7 & 8 on ResNet-34 [18], PreActResNet-34.
TanhSoft—Dynamic Trainable Activation Functions

FIGURE 9. Top-1 train and test accuracy (higher is better) on CIFAR100 dataset with WideResNet 28-10 model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

FIGURE 10. Top-1 train and test loss (lower is better) on CIFAR100 dataset with WideResNet 28-10 model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

FIGURE 11. Top-1 train and test accuracy (higher is better) on CIFAR10 dataset with LeNet model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

FIGURE 12. Top-1 train and test loss (lower is better) on CIFAR10 dataset with LeNet model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

(36x752) K. Biswas et al.: TanhSoft—Dynamic Trainable Activation Functions

FIGURE 9. Top-1 train and test accuracy (higher is better) on CIFAR100 dataset with WideResNet 28-10 model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

FIGURE 10. Top-1 train and test loss (lower is better) on CIFAR100 dataset with WideResNet 28-10 model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

FIGURE 11. Top-1 train and test accuracy (higher is better) on CIFAR10 dataset with LeNet model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

FIGURE 12. Top-1 train and test loss (lower is better) on CIFAR10 dataset with LeNet model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

PA-ResNet-34) [41], VGG-16 (with Batch-normalization) [19], Densenet-121 (DN-121) [20], DenseNet-169 (DN-169) [20], InceptionNet V3 (IN-V3) [22], SimpleNet(SN) [42], MobileV2 (MN) [21], WideResNet 28-10 (WRN 28-10) [43], GoogleNet (GN) [44], ResNeXt-50 [45], StochasticDepth [46], Shufflenet V2 [47], Deep Layer Aggregation (DLA) [48], RegNet [49], NASNet [50], ResNet in ResNet (RIR) [51], Xception Network [52], EfficientNet B0 (EN-B0) [53], Le-Net [54] and SqueezeNet (SQ-Net) [55] models. It is clear from these tables that TanhSoft-1, TanhSoft-2, and TanhSoft-3 constantly outperforms ReLU and Swish in most cases and we have got around 1% to 6% improvement in Top-1 accuracy when compared to models with ReLU activation. We have trained the networks with batch size 128, Adam optimizer [56] with 0.001 learning rate and up to 100 epochs for all the models mentioned above except SimpleNet and VGG-16, which is trained till 200 epochs. Data augmentation is used for both datasets. Learning curves of ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3 activations are given in Figures 9 & 10 on WideResNet 28-10 model in CIFAR100 dataset and Figures 11 & 12 on Le-net model in CIFAR10 dataset. From these figures it is evident that after training few epochs TanhSoft-1, TanhSoft-2, TanhSoft-3 have faster convergence capability, higher accuracy and lower loss when compared to ReLU.

5) TINY IMAGENET

The ImageNet Large Scale Visual Recognition Challenge(ILSVRC) is one of the most popular benchmarks for image classification problems. Tiny ImageNet Challenge is a similar type of challenges like ILSVRC for image classification, which has a smaller dataset and fewer image classes. The database has images of size $64 \times 64$ with 200 image
classes with a training dataset of 100,000 images, a validation dataset of 10,000 images, and a test dataset of 10,000 images. Each class has 500 training images, 50 validation images, and 50 test images. We have reported results for top-1 accuracy for mean of 5 runs in Table 9 on WideResNet 28-10 (WRN 28-10) [43] model. The network is trained with the Normal initializer [7], a batch size of 32, Adam optimizer [56], 0.2 dropout rate [40], initial learning rate (lr rate) 0.01, and reduce lr rate by a factor of 10 after every 50 epochs up to 250 epochs. We have used data augmentation method in this database.

B. OBJECT DETECTION

Object Detection is an important problem in computer vision. We have considered the Pascal VOC dataset [57] for our experiments. Results are reported on Single Shot MultiBox Detector (SSD) 300 model [58], and VGG-16 (with batch-normalization) considered as the backbone network. The model is trained on Pascal VOC 07+12 training data, and model performance is evaluated on Pascal VOC 2007 test data. The model has been trained with a batch size of 8, $5e^{-4}$ weight decay for 120000 iterations, 0.001 learning rate, SGD optimizer [59], [60] with 0.9 momentum. No pre-trained weight is used in the network. The mean average precision (mAP) is reported in Table 10 for a mean of 5 different runs.

C. SEMANTIC SEGMENTATION

Semantic segmentation is a very important problem in computer vision. We have shown our experimental results on the CityScapes dataset [61]. CityScapes training data with U-net model [62] is trained for 250 epochs, with adam optimizer [56], with batch size 32 and Xavier Uniform initializer [63], and learning rate $5e^{-3}$. Pixel Accuracy and mean Intersection-Over-Union (mIOU) on test data have been reported on Table 11 for mean of 5 different runs.

D. MACHINE TRANSLATION

Machine Translation is a deep learning technique to translate from one language to another. For this problem, WMT 2014 English→German dataset is used. It has 4.5 million training sentences, and model performance is evaluated on the newstest2014 dataset using the BLEU score metric. We have consider an Attention-based multi-head transformer model [64] for our experiments. A 8-head transformer model is considered with 0.1 dropout [40], Adam optimizer [56], and trained for 100000 steps. Other hyper-parameters are tried to retain similar as mentioned in the original paper [64]. We have reported a Mean of 5 runs has on Table 12 on the test dataset(newstest2014).

E. COMPARISON WITH BASELINES

Based on all the experiments given in earlier sections, we observe that TanhSoft-1, TanhSoft-2, and TanhSoft-3, beats or performs equally well in most cases when compared with the baseline activation functions and under-performs marginally on rare occasions, and we provide a detailed comparison of the proposed activations with the baseline activations in Table 13. Table 13 contains the total number

| Activation Function | Pixel Accuracy | mIOU |
|---------------------|----------------|------|
| TanhSoft-1          | 80.71          | 70.45|
| TanhSoft-2          | 80.62          | 70.34|
| TanhSoft-3          | 80.65          | 70.37|
| ReLU                | 79.54          | 69.39|
| Swish               | 79.87          | 69.68|
| Leaky ReLU($\alpha = 0.01$) | 79.59      | 69.48|
| ELU                 | 79.12          | 68.12|
| Softplus            | 78.89          | 68.04|
| Mish                | 80.39          | 69.87|
| GELU                | 79.69          | 69.62|

TABLE 9. Experimental results on tiny ImageNet dataset. Mean of 5 different runs for top-1 accuracy (in %) have been reported.

| Activation Function | Wide ResNet 28-10 Model |
|---------------------|------------------------|
| TanhSoft-1          | 62.61 |
| TanhSoft-2          | 62.28 |
| TanhSoft-3          | 62.17 |
| ReLU                | 60.35 |
| Swish               | 60.69 |
| Leaky ReLU($\alpha = 0.01$) | 60.62 |
| ELU                 | 60.02 |
| Softplus            | 59.81 |
| Mish                | 60.77 |
| GELU                | 60.72 |

TABLE 10. Object detection results on SSD 300 model in Pascal-VOC dataset.

| Activation Function | mAP |
|---------------------|-----|
| TanhSoft-1          | 77.9 |
| TanhSoft-2          | 78.0 |
| TanhSoft-3          | 77.8 |
| ReLU                | 77.2 |
| Swish               | 77.3 |
| Leaky ReLU($\alpha = 0.01$) | 77.2 |
| ELU                 | 75.1 |
| Softplus            | 74.2 |
| Mish                | 77.5 |
| GELU                | 77.3 |

TABLE 11. Semantic segmentation results on U-NET model in Cityscapes dataset.

| Activation Function | mAP |
|---------------------|-----|
| TanhSoft-1          | 26.7 |
| TanhSoft-2          | 26.7 |
| TanhSoft-3          | 26.6 |
| ReLU                | 26.2 |
| Swish               | 26.4 |
| Leaky ReLU($\alpha = 0.01$) | 26.3 |
| ELU                 | 25.1 |
| Softplus            | 23.6 |
| Mish                | 26.3 |
| GELU                | 26.2 |

TABLE 12. Machine translation results on transformer model in WMT-2014 dataset.
of cases in which the proposed activations performs better, equal or less than the baseline activations. The proposed activations outperform concerning model performance in all cases compared to ReLU, Leaky ReLU, ELU, and Softplus. Also, compared to Swish, Mish, and GELU, the proposed activations outperform most cases while under-performing on infrequent occasions.

F. COMPUTATIONAL TIME COMPARISON

We have reported the computational time comparison for TanhSoft-1, TanhSoft-2, and TanhSoft-3 and the baseline activation functions for both forward and backward pass on a $32 \times 32$ RGB image in ResNet-34 model in Table 14 for the mean of 100 runs. All the runs are performed on an NVIDIA Tesla V100 GPU with 16GB ram. The computational time for both forward and backward passes are reported in milliseconds ($\mu$s). From Table 14, we notice that due to the nonliterary of the proposed activations, the computational time for both forward and backward pass is slightly higher than ReLU ($\mu$s) while it is similar to Mish and better than GELU. Also, due to the non-linearity of the proposed activations, there is a trade-off between state-of-the-art model performances and computational time. From the experimental section, we notice that compared to ReLU networks with the proposed activations networks, the model performance has increased significantly, but the computational time increased marginally.

V. CONCLUSION

We have explored three novel trainable activation functions in this work, TanhSoft-1, TanhSoft-2, and TanhSoft-3. The proposed functions are zero-centred, non-monotonic, non-zero negative bounded curve, continuous, and differentiable. In the beginning, we have conducted experiments with the three activations with constant hyper-parameters, and we found that these activations perform equally or slightly better than ReLU. Later, we tune the hyper-parameters via backpropagation and make the proposed activations trainable. In this case, we found a considerable change in results (Top-1 accuracy or mAP or mIOU or BLEU score), and they perform far better than ReLU or the other baseline activations in most of the experiments. It shows that introducing a trainable parameter plays an essential role in activation functions, and a non-zero bounded negative part & trainable parameters result in better performance. We have used hyperparameters and models with the ReLU activation function and then replace ReLU with other baseline activations & the proposed activations to compare model performances. Our empirical evaluation on different deep learning tasks like Image classification, Object Detection, Semantic Segmentation, Machine Translation in a variety of complex models on datasets like MNIST, Fashion MNIST, SVHN, CIFAR10, CIFAR100, Tiny ImageNet, Pascal VOC, CityScapes, and WMT 2014 shows that the proposed activation functions produce state-of-the-art results and have an excellent potential to replace the widely used activation functions like ReLU, Leaky ReLU, ELU, Softplus, Swish, Mish, and GELU.

REFERENCES

[1] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” 2018, arXiv:1811.03378 [Online]. Available: https://arxiv.org/abs/1811.03378
[2] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in Proc. 27th Int. Conf. Mach. Learn. (ICML), J. Fürnkranz and T. Joachims, Eds. Haifa, Israel: Omnipress, Jun. 2010, pp. 807–814.
[3] R. Hahnloser, R. Sarpeshkar, M. Mahowald, R. Douglas, and H. Seung, “Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit,” Nature, vol. 405, pp. 947–951, Jul. 2000.
[4] Y. Zhou, D. Li, S. Hua, and S.-Y. Kung, “Soft-root-sign activation function,” Tech. Rep., 2020.
K. Biswas et al.: TanhSoft—Dynamic Trainable Activation Functions

[59] H. Robbins and S. Monro, “A stochastic approximation method.” Ann. Math. Statist., vol. 22, no. 3, pp. 400–407, Sep. 1951.

[60] J. Kiefer and J. Wolfowitz, “Stochastic estimation of the maximum of a regression function,” Ann. Math. Stat., vol. 23, no. 3, pp. 462–466, 1952.

[61] M. Cords, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 3213–3223.

[62] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” Tech. Rep., 2015.

[63] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in Proc. 13th Int. Conf. Artif. Intell. Statist., vol. 9, Y. W. Teh and M. Titterington, Eds. Sardinia, Italy: Chia Laguna Resort, May 2010, pp. 249–256.

[64] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, N. A. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5998–6008.

KOUSHIK BISWAS is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, IIIT Delhi, India. His research interests include deep learning and computer vision.

SANDEEP KUMAR is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, IIIT Delhi, India. He is currently working as an Assistant Professor with the Department of Mathematics, Shaheed Bhagat Singh College, University of Delhi, India. His research interests include deep learning and computer vision.

SHILPAK BANERJEE received the Ph.D. degree in mathematics from Pennsylvania State University, in 2017. He is currently working as an Assistant Professor with the Department of Mathematics, IIIT Delhi, India. His research interests include dynamical systems and ergodic theory.

ASHISH KUMAR PANDEY received the Ph.D. degree in mathematics from the University of Illinois at Urbana–Champaign, Urbana–Champaign, USA, in April 2018. He is currently working as an Assistant Professor with the Department of Mathematics, IIIT Delhi, India. His research interests include partial differential equations and machine learning.

KOUSHIK BISWAS is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, IIIT Delhi, India. His research interests include deep learning and computer vision.

ASHISH KUMAR PANDEY received the Ph.D. degree in mathematics from the University of Illinois at Urbana–Champaign, Urbana–Champaign, USA, in April 2018. He is currently working as an Assistant Professor with the Department of Mathematics, IIIT Delhi, India. His research interests include partial differential equations and machine learning.

* * *