SAU: Smooth activation function using convolution with approximate identities

Koushik Biswas[1][0000−0002−9818−8966], Sandeep Kumar[1,3][0000−0002−5464−929X], Shilpak Banerjee[2][0000−0003−1036−9576], and Ashish Kumar Pandey[4][0000−0002−9723−035X]

[1] Department of Computer Science, IIIT Delhi, New Delhi, India.
{koushikb,sandeepk}@iiitd.ac.in
[2] Department of Mathematics and Statistics, IIT Tirupati, India.
shilpak@iittp.ac.in
[3] Department of Mathematics, Shaheed Bhagat Singh College, University of Delhi, New Delhi, India.
[4] Department of Mathematics, IIIT Delhi, New Delhi, India.
ashish.pandey@iiitd.ac.in

Abstract. Well-known activation functions like ReLU or Leaky ReLU are non-differentiable at the origin. Over the years, many smooth approximations of ReLU have been proposed using various smoothing techniques. We propose new smooth approximations of a non-differentiable activation function by convolving it with approximate identities. In particular, we present smooth approximations of Leaky ReLU and show that they outperform several well-known activation functions in various datasets and models. We call this function Smooth Activation Unit (SAU). Replacing ReLU by SAU, we get 5.63%, 2.95%, and 2.50% improvement with ShuffleNet V2 (2.0x), PreActResNet 50 and ResNet 50 models respectively on the CIFAR100 dataset and 2.31% improvement with ShuffleNet V2 (1.0x) model on ImageNet-1k dataset.

Keywords: Smooth Activation Function, Neural Network.

1 Introduction

Deep networks form a crucial component of modern deep learning. Non-linearity is introduced in such networks by the use of activation functions, and the choice has a substantial impact on network performance and training dynamics. Designing a new novel activation function is a difficult task. Handcrafted activations like Rectified Linear Unit (ReLU) ([32]), Leaky ReLU ([29]) or its variants are very common choices for activation functions and exhibits promising performance on different deep learning tasks. There are many activations that have been proposed so far and some of them are ELU ([3]), Parametric ReLU (PReLU) ([8]), Swish ([36]), Padé Activation Unit (PAU) ([31], ACON [27], Mish ([30]), GELU ([10]), ReLU6 ([20]), Softplus ([50]) etc. Nevertheless, ReLU remains the favourite choice among the deep learning community due to its simplicity and

better performance when compared to Tanh or Sigmoid, though it has a drawback known as dying ReLU, in which the network starts to lose the gradient direction due to the negative inputs and produces zero outcome. In 2017, Swish ([36]) was proposed by the Google brain team. Swish was found by automatic search technique, and it has shown some promising performance across different deep learning tasks.

Activation functions are usually handcrafted. PReLU ([8]) tries to overcome this problem by introducing a learnable negative component to ReLU ([32]). Maxout ([6]) and Mixout ([49]) are constructed with piecewise linear components, and theoretically, they are universal function approximators, though they increase the number of parameters in the network. Recently, meta-ACON ([27]), a smooth activation, has been proposed, which is the generalization of the ReLU and Maxout activations and can smoothly approximate Swish. Meta-ACON has shown some good improvement on both small models and highly optimized large models. PAU ([31]) is a promising candidate for trainable activations, which have been introduced recently based on rational function approximation.

In this paper, we introduce a smooth approximation of known non-smooth activation functions like ReLU or Leaky ReLU based on the approximation of identity. Our experiments show that the proposed activations improve the performance of different network architectures compared to ReLU on different deep learning problems.

2 Mathematical formalism

2.1 Convolution

Convolution is a binary operation, which takes two functions f and g as input, and outputs a new function denoted by $f * g$. Mathematically, we define this operation as follows

$$(f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x-y)\,dy. \tag{1}$$

The convolution operation has several properties. Below, we will list two of them which will be used later in this article.

P1. $(f * g)(x) = (g * f)(x)$,
P2. If f is n-times differentiable with compact support over $\mathbb{R}$ and g is locally integrable over $\mathbb{R}$ then $f * g$ is at least n-times differentiable over $\mathbb{R}$.

Property P1 is an easy consequence of definition (1). Property P2 can be easily obtained by moving the derivative operator inside the integral. Note that this exchange of derivative and integral requires f to be of compact support. An immediate consequence of property P2 is that if one of the functions f or g is smooth with compact support, then $f*g$ is also smooth. This observation will be used later in the article to obtain smooth approximations of non-differentiable activation functions.

2.2 Mollifier and Approximate identities

A smooth function ϕ over $\mathbb{R}$ is called a mollifier if it satisfies the following three properties:

1. It is compactly supported.
2. $\int_{\mathbb{R}} \phi(x)\,dx = 1$.
3. $\lim_{\epsilon \to 0} \phi_\epsilon(x) := \lim_{\epsilon \to 0} \frac{1}{\epsilon}\phi(x/\epsilon) = \delta(x)$, where $\delta(x)$ is the Dirac delta function.

We say that a mollifier ϕ is an approximate identity if for any locally integrable function f over $\mathbb{R}$, we have

$$\lim_{\epsilon \to 0}(f * \phi_\epsilon)(x) = f(x) \text{ pointwise for all } x.$$

2.3 Smooth approximations of non-differentiable functions

Let ϕ be an approximate identity. Choosing $\epsilon = 1/n$ for $n \in \mathbb{N}$, one can define

$$\phi_n(x) := n\phi(nx). \tag{2}$$

Using the property of approximate identity, for any locally integrable function f over $\mathbb{R}$, we have

$$\lim_{n \to \infty}(f * \phi_n)(x) = f(x) \text{ pointwise for all } x.$$

That is, for large enough n, $f * \phi_n$ is a good approximation of f. Moreover, since ϕ is smooth, ϕ_n is smooth for each $n \in \mathbb{N}$ and therefore, using property P2, $f * \phi_n$ is a smooth approximation of f for large enough n.

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be any activation function. Then, by definition, σ is a continuous and hence, a locally integrable function. For a given approximate identity ϕ and $n \in \mathbb{N}$, we define a smooth approximation of σ as $\sigma * \phi_n$, where ϕ_n is defined in (2).

3 Smooth Activation Unit (SAU)

Consider the Gaussian function

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

which is a well-known approximate identity. Consider the Leaky Rectified Linear Unit (Leaky ReLU) activation function

$$\text{LeakyReLU}[\alpha](x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases}$$

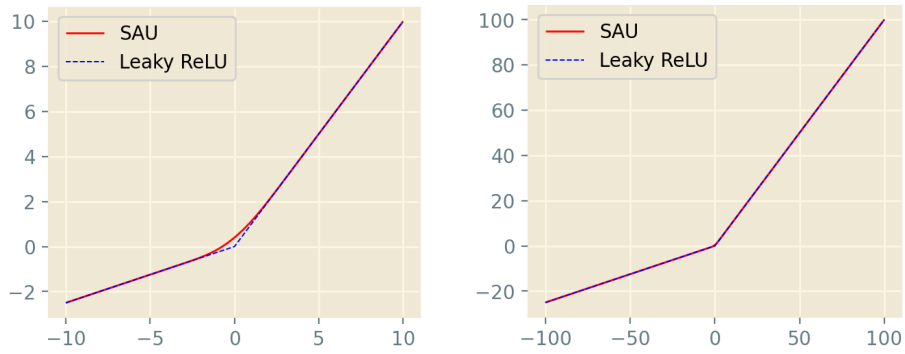

Fig. 1: Approximation of Leaky ReLU ($\alpha = 0.25$) using SAU. The left figure shows that SAU approximates Leaky ReLU smoothly, and in the right figure, we plot the same functions on a larger domain range.

Note that LeakyReLU[α] activation function is hyperparametrized by α and it is non-differentiable at the origin for all values of α except $\alpha = 1$. For $\alpha = 0$, LeakyReLU[α] reduces to well known activation function ReLU ([32]) while for constant and trainable α, LeakyReLU[α] reduces to Leaky ReLU ([29]) and Parametric ReLU ([8]) respectively. For a given $n \in \mathbb{N}$, and $\alpha \neq 1$, a smooth approximation of LeakyReLU[α] is given by

$$G(x, \alpha, n) = (\text{LeakyReLU}[\alpha] * \phi_n)(x) = \frac{1}{2n}\sqrt{\frac{2}{\pi}} e^{\frac{-n^2x^2}{2}} + \frac{(1+\alpha)}{2}x + \frac{(1-\alpha)}{2}x \ \text{erf}\left(\frac{nx}{\sqrt{2}}\right) \tag{3}$$

where erf is the Gaussian error function

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt.$$

For the rest of the paper, we will only consider the approximate identity of Leaky ReLU given in (3) as the activation function. We call this function Smooth Activation Unit (SAU). Approximation of Leaky ReLU ($\alpha = 0.25$) by SAU is given in figure 1. It is clear from the figure 1 that SAU can smoothly approximate Leaky ReLU (as well as ReLU or its variants) quite well. We note that in GELU ([10]) paper, the authors use the product of x with the cumulative distribution function of a suitable probability distribution (see ([10]) for further details).

3.1 Learning activation parameters via back-propagation

Back-propagation algorithm ([22]) and gradient descent is used in neural networks to update Weights and biases. Parameters in trainable activation functions

are updated using the same technique. The forward pass is implemented in both Pytorch ([35]) & Tensorflow-Keras ([1]) API and the parameters are updated by automatic differentiation. Alternatively, CUDA ([34]) can be used to implement (see ([29])) the gradients of equation 3 for the input x and the parameters α & n and it can be computed as follows:

$$\frac{\partial G}{\partial x} = \frac{-nx}{2}\sqrt{\frac{2}{\pi}}e^{\frac{-n^2x^2}{2}} + \frac{(1+\alpha)}{2} + \frac{(1-\alpha)}{2}\ \text{erf}\left(\frac{nx}{\sqrt{2}}\right) + \frac{n(1-\alpha)}{\sqrt{2\pi}}\ x\ e^{-\frac{n^2x^2}{2}} \tag{4}$$

$$\frac{\partial G}{\partial \alpha} = \frac{x}{2}\left(1 - \text{erf}\left(\frac{nx}{\sqrt{2}}\right)\right). \tag{5}$$

$$\frac{\partial G}{\partial n} = -\frac{1}{2n^2}\sqrt{\frac{2}{\pi}}e^{\frac{-n^2x^2}{2}} - \frac{x^2}{2}\sqrt{\frac{2}{\pi}}e^{\frac{-n^2x^2}{2}} + \frac{x^2(1-\alpha)}{\sqrt{2\pi}}\ e^{-\frac{n^2x^2}{2}}. \tag{6}$$

where

$$\frac{d}{dx}\text{erf}(x) = \frac{2}{\sqrt{\pi}}e^{-x^2}$$

α and n can be either hyperparameters or trainable parameters.
Now, note that the class of neural networks with SAU activation function is dense in $C(K)$, where K is a compact subset of $\mathbb{R}^n$ and $C(K)$ is the space of all continuous functions over K.
The proof follows from the following proposition (see ([31])).

Proposition 1. (Theorem 1.1 in Kidger and Lyons, 2020 ([16])) :- Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous function. Let N_n^ρ represent the class of neural networks with activation function ρ, 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_n^ρ is dense in $C(K)$ if and only if ρ is non-polynomial.

4 Experiments

To explore and compare the performance of SAU, we consider eight popular standard activation functions on different standard datasets and popular network architectures on standard deep learning problems like image classification, object detection, semantic segmentation, and machine translation. We consider the following activations to compare with SAU: ReLU, Leaky ReLU, Parametric ReLU (PReLU), ELU, ReLU6, Softplus, PAU, Swish, and GELU. It is evident from the experimental results in the next sections that SAU outperform in most cases compared to the standard activations. We consider α as a hyperparameter and n as a trainable parameter for the rest of our experiments. The value of n is initialised at 20000 and updated via backpropagation according to equation 6. To run the experiments, we use an NVIDIA Tesla V100 GPU with 32GB RAM.

4.1 Image Classification

MNIST, Fashion MNIST and The Street View House Numbers (SVHN) Database: In this section, we present results on MNIST ([24]), Fashion MNIST ([45]), and SVHN ([33]) datasets. The MNIST and Fashion MNIST databases have a total of 60k training and 10k testing 28×28 grey-scale images with ten different classes. SVHN consists of 32×32 RGB images with a total of 73257 training images and 26032 testing images with ten different classes. We have applied standard data augmentation methods like rotation, zoom, height shift, and shearing on the three datasets. We report results with LeNet [23], AlexNet ([21]), and VGG-16 ([40]) (with batch-normalization ([15])) architecture in Table 1, Table 2, and Table 3 respectively. For all the experiments to train a model on these three datasets, we use a batch size of 128, stochastic gradient descent ([37], [17]) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 100 epochs. We begin with 0.01 learning rate and decay the learning rate with the cosine annealing ([26]) learning rate scheduler. We report more experiments on these datasets with a custom-designed model in the supplementary material.

Table 1: A Detailed Comparison between SAU Activation Function and Other Baseline Activation Functions on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with the LeNet Architecture. top-1 Test Accuracy (in %) is Reported in the Table for the Mean of 10 Different Runs. We Report mean±std in the Table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.21 ± 0.10	91.51 ± 0.20	92.17 ± 0.19
Leaky ReLU	99.17 ± 0.10	91.61 ± 0.21	92.31 ± 0.18
PReLU	99.27 ± 0.09	91.62 ± 0.18	92.05 ± 0.21
ReLU6	99.29 ± 0.08	91.57 ± 0.17	92.25 ± 0.17
ELU	99.28 ± 0.10	91.48 ± 0.19	92.20 ± 0.18
Softplus	99.06 ± 0.16	91.21 ± 0.23	91.89 ± 0.25
PAU	99.34 ± 0.07	**91.69** ± 0.12	92.31 ± 0.22
Swish	99.31 ± 0.07	91.64 ± 0.14	92.39 ± 0.20
GELU	99.29 ± 0.06	91.61 ± 0.14	92.42 ± 0.20
SAU	**99.40** ± 0.05	91.47 ± 0.16	**92.61** ± 0.12

CIFAR: The CIFAR ([19]) is one of the most popular databases for image classification problem consists of a total of 60k 32×32 RGB images and is divided into 50k training and 10k test images. CIFAR has two different datasets- CIFAR10 and CIFAR100 with a total of 10 and 100 classes, respectively. We report the top-1 accuracy on Table 6 and Table 4 on CIFAR10 and CIFAR100 datasets respectively. We consider MobileNet V1 ([11]), MobileNet V2 ([39]), Shufflenet

V2 ([28]), PreActResNet ([9]), ResNet ([7]), Inception V3 ([42]), squeeze and excitation networks (SeNet) [12], ResNext [46], LeNet [23], AlexNet [21], DenseNet ([13]), Xception [2], Squeezenet [14], WideResNet ([47]), VGG ([40]) (with batch-normalization ([15])), and EfficientNet B0 ([43]). For all the experiments to train a model on these two datasets, we use a batch size of 128, stochastic gradient descent ([37], [17]) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 200 epochs. We begin with 0.01 learning rate and decay the learning rate by a factor of 10 after every 60 epochs. Standard data augmentation methods like horizontal flip and rotation are applied to both datasets. It is noticeable from these two tables that by replacing ReLU with SAU, there is an increment in top-1 accuracy from 1% to more than 5% in most of the models. More detailed results on these two datasets with other baseline activations are reported in the supplementary material. Training and test accuracy & loss curves for baseline activation functions and SAU are given in Figures 2 and 3 respectively on CIFAR100 dataset on ShuffleNet V2 (2.0x) network. From these learning curves, it is evident that after training a few epochs, SAU has stable & smooth learning and higher accuracy and lower loss on the test dataset compared to other baseline activation functions.

Table 2: A Detailed Comparison between SAU Activation Function and Other Baseline Activation Functions on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with the Alexnet Architecture. top-1 Test Accuracy (in %) is Reported in the Table for the Mean of 10 Different Runs. We Report mean±std in the Table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.51 ± 0.06	92.77 ± 0.18	95.11 ± 0.14
Leaky ReLU	99.50 ± 0.06	92.79 ± 0.20	95.21 ± 0.17
PReLU	99.48 ± 0.08	92.76 ± 0.18	95.19 ± 0.17
ReLU6	99.55 ± 0.06	93.01 ± 0.16	95.22 ± 0.15
ELU	99.56 ± 0.05	92.89 ± 0.17	95.30 ± 0.18
Softplus	99.22 ± 0.10	92.32 ± 0.25	94.82 ± 0.21
PAU	99.53 ± 0.08	93.01 ± 0.17	95.22 ± 0.13
Swish	99.58 ± 0.06	92.96 ± 0.16	95.32 ± 0.14
GELU	99.55 ± 0.06	93.05 ± 0.14	95.28 ± 0.14
SAU	$\mathbf{99.64} \pm 0.04$	$\mathbf{93.17} \pm 0.14$	$\mathbf{95.45} \pm 0.11$

Also, We compare the performance of SAU with other baseline activations with state-of-the-art data augmentation methods like Mixup [48] on CIFAR 100 dataset with ShuffleNet V2 (2.0x), ResNet 18 & ResNet 50 models, and we got very good improvement over the baseline activations. Results are reported in Table 5 for the mean of 10 different runs. We use the same experimental setup as used for the CIFAR100 dataset.

Table 3: A Detailed Comparison between SAU Activation Function and Other Baseline Activation Functions on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with the VGG16 Architecture. top-1 Test Accuracy (in %) is Reported in the Table for the Mean of 10 Different Runs. We Report mean±std in the Table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.55 ± 0.07	93.75 ± 0.14	96.04 ± 0.12
Leaky ReLU	99.59 ± 0.05	93.89 ± 0.14	96.12 ± 0.15
PReLU	99.58 ± 0.07	93.85 ± 0.16	96.12 ± 0.17
ReLU6	99.59 ± 0.05	93.88 ± 0.11	96.18 ± 0.16
ELU	99.51 ± 0.05	93.82 ± 0.16	96.13 ± 0.14
Softplus	99.34 ± 0.12	93.69 ± 0.19	95.88 ± 0.21
PAU	99.58 ± 0.05	94.27 ± 0.12	96.20 ± 0.15
Swish	99.54 ± 0.06	94.10 ± 0.12	96.26 ± 0.13
GELU	99.60 ± 0.04	94.17 ± 0.12	96.23 ± 0.13
SAU	**99.67** ± 0.04	**94.40** ± 0.12	**96.41** ± 0.12

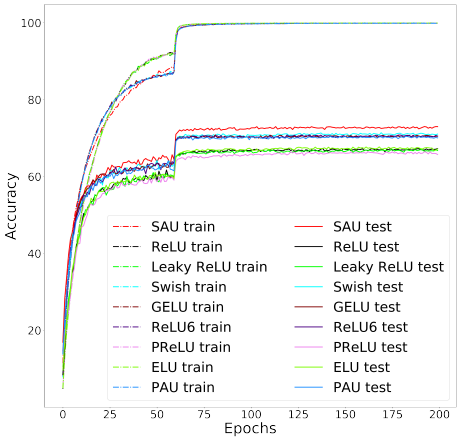

Fig. 2: Top-1 Train and Test accuracy Curves (Higher is Better) for SAU and Baseline Activation Functions on CIFAR100 Dataset with ShuffleNet V2 (2.0x) Model.

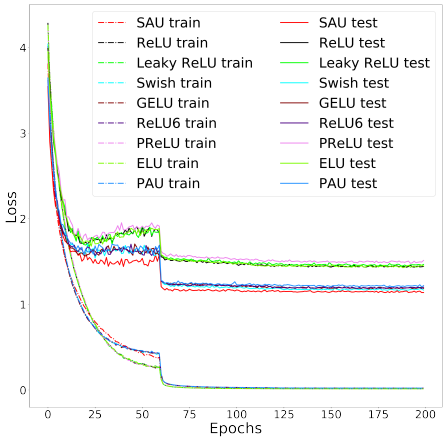

Fig. 3: Top-1 Train and Test Loss Curves (Lower is Better) for SAU and Baseline Activation Functions on CIFAR100 Dataset with ShuffleNet V2 (2.0x) Model.

Table 4: A Detailed Comparison between SAU Activation and Other Baseline Activation Functions (See Supplementary Document for More Detailed Experimental Results) on the CIFAR100 Dataset for Image Classification Problem with Different Popular Network Architectures. top-1 Test Accuracy (in %) is Reported in the Table for the Mean of 10 Different Runs. We Report mean±std in the Table.

Model	ReLU	SAU
	Top-1 accuracy (mean± std)	Top-1 accuracy (mean ± std)
Shufflenet V2 0.5x	61.76 ± 0.27	**64.39** ± 0.23
Shufflenet V2 1.0x	64.12 ± 0.28	**68.41** ± 0.24
Shufflenet V2 1.5x	66.52 ± 0.28	**71.97** ± 0.24
Shufflenet V2 2.0x	66.94 ± 0.24	**72.57** ± 0.21
PreActResNet 18	72.58 ± 0.24	74.01 ± 0.22
PreActResNet 34	72.92 ± 0.24	**75.37** ± 0.24
PreActResNet 50	73.27 ± 0.25	**76.22** ± 0.22
ResNet 18	73.02 ± 0.25	74.27 ± 0.22
ResNet 34	73.12 ± 0.26	**74.64** ± 0.23
ResNet 50	73.89 ± 0.23	**76.39** ± 0.20
MobileNet V1	70.95 ± 0.26	**72.09** ± 0.23
MobileNet V2	73.85 ± 0.24	**75.69** ± 0.19
Inception V3	74.03 ± 0.27	**76.01** ± 0.22
WideResNet 28-10	75.89 ± 0.23	**77.39** ± 0.20
DenseNet 121	75.72 ± 0.27	**77.11** ± 0.23
EffitientNet B0	76.22 ± 0.24	**78.07** ± 0.26
VGG16	71.10 ± 0.30	71.18 ± 0.28

Table 5: Top-1 Test Accuracy Reported with Mixup Augmentation Method on CIFAR100 Dataset for the Mean of 10 Different Runs. We Report mean±std in the Table.

Activation Function	ShuffleNet V2 (2.0x)	ResNet 50	ResNet 18
ReLU	69.10 ± 0.24	75.10 ± 0.23	73.88 ± 0.24
Leaky ReLU	69.04 ± 0.23	75.04 ± 0.23	73.97 ± 0.26
PReLU	69.29 ± 0.25	75.17 ± 0.25	74.12 ± 0.25
ReLU6	69.36 ± 0.23	75.27 ± 0.22	74.17 ± 0.23
ELU	69.34 ± 0.24	75.32 ± 0.24	74.03 ± 0.24
Softplus	68.84 ± 0.28	74.52 ± 0.26	73.69 ± 0.27
Swish	72.78 ± 0.21	76.42 ± 0.22	74.39 ± 0.23
GELU	72.91 ± 0.22	76.54 ± 0.23	74.51 ± 0.23
PAU	73.09 ± 0.22	76.77 ± 0.22	74.62 ± 0.25
SAU	**74.22** ± 0.21	**77.81** ± 0.21	**75.59** ± 0.21

Table 6: A Detailed Comparison between SAU Activation and Other Baseline Activation Functions (See Supplementary Document for More Detailed Experimental Results) on the CIFAR10 Dataset for Image Classification Problem with Different Popular Network Architectures. top-1 Test Accuracy (in %) is Reported in the Table for the Mean of 10 Different Runs. We Report mean±std in the Table.

Model	ReLU	SAU
	Top-1 accuracy (mean± std)	Top-1 accuracy (mean ± std)
ShuffleNet V2 0.5x	88.01 ± 0.23	**90.50** ± 0.17
ShuffleNet V2 1.0x	90.74 ± 0.25	**92.78** ± 0.20
ShuffleNet V2 1.5x	91.07 ± 0.23	**93.20** ± 0.18
ShuffleNet V2 2.0x	91.32 ± 0.22	**93.52** ± 0.16
PreActResNet 18	93.36 ± 0.18	94.62 ± 0.15
PreActResNet 34	94.01 ± 0.16	**95.10** ± 0.14
PreActResNet 50	94.01 ± 0.15	**94.94** ± 0.14
ResNet 18	93.32 ± 0.20	93.47 ± 0.17
ResNet 34	93.77 ± 0.20	**94.22** ± 0.16
ResNet 50	93.89 ± 0.19	**94.62** ± 0.16
MobileNet V1	92.27 ± 0.24	**93.54** ± 0.14
MobileNet V2	93.89 ± 0.19	**95.37** ± 0.09
Inception V3	93.89 ± 0.18	**94.51** ± 0.10
WideResNet 28-10	94.74 ± 0.18	**95.52** ± 0.12
DenseNet 121	94.41 ± 0.16	**95.31** ± 0.10
EffitientNet B0	94.64 ± 0.16	**95.52** ± 0.14
VGG16	93.14 ± 0.23	93.31 ± 0.21

Tiny Imagenet: This section presents results on the Tiny ImageNet dataset, a similar kind of image classification database to the ImageNet Large Scale Visual Recognition Challenge(ILSVRC). Tiny Imagenet dataset contains 64×64 RGB images with total 100,000 training images, 10,000 validation images, and 10,000 test images and have total 200 image classes. We report the mean of 6 different runs for Top-1 accuracy in table 7 on WideResNet 28-10 (WRN 28-10) ([47]) and ResNet 18 [7] models. We consider a batch size of 64, 0.2 dropout rate ([41]), SGD optimizer ([37], [17]), He Normal initializer ([8]), initial learning rate(lr rate) 0.1, and lr rate is reduced by a factor of 10 after every 50 epochs up-to 300 epochs. Standard data augmentation techniques like rotation, width shift, height shift, shearing, zoom, horizontal flip, and fill mode are applied to improve performance. It is evident from the table that the proposed function performs better than the baseline functions, and top-1 accuracy is stable (mean±std) and got a good improvement for SAU over ReLU.

Table 7: A Detailed Comparison between SAU Activation Function and Other Baseline Activation Functions on Tiny ImageNet Dataset for Image Classification Problem. top-1 Test Accuracy (in %) is Reported in the Table for the Mean of 6 Different Runs. We Report mean±std in the Table.

Activation Function	WideResNet 28-10	ResNet 18
ReLU	62.77 ± 0.46	58.27 ± 0.42
Leaky ReLU	62.72 ± 0.46	58.52 ± 0.44
PReLU	62.70 ± 0.48	58.39 ± 0.44
ReLU6	62.59 ± 0.46	58.67 ± 0.41
ELU	62.58 ± 0.50	58.62 ± 0.43
Softplus	61.77 ± 0.59	58.04 ± 0.47
PAU	63.62 ± 0.44	59.47 ± 0.40
Swish	63.47 ± 0.46	59.02 ± 0.42
GELU	63.26 ± 0.48	59.27 ± 0.39
SAU	**64.07** ± 0.44	**60.12** ± 0.40

ImageNet-1k: ImageNet-1k is a popular image database with more than 1.2 million training images with 1000 classes. We report result on ImageNet-1k with ShuffleNet V2 [28] and ResNet-50 [7] model in Table 8. We use a batch size of 256, SGD optimizer ([37], [17]), 0.9 momentum, and $5e^{-4}$ weight decay. We consider a linear decay learning rate scheduler from 0.1 and trained upto 600k iterations. Experiments on ImageNet-1k are conducted on four NVIDIA V100 GPUs with 32GB RAM each.

Table 8: A Detailed Comparison between SAU Activation Function and Other Baseline Activation Functions on ImageNet-1k Dataset for Image Classification Problem. We Report top-1 Accuracy in the Table.

Activation Function	ShuffleNet V2 (1.0x)	ResNet-50
ReLU	69.31	75.50
Leaky ReLU	69.25	75.64
PReLU	69.20	75.48
ReLU6	69.44	75.77
ELU	69.62	75.54
Softplus	69.21	75.37
Swish	70.71	76.39
GELU	70.31	76.30
PAU	70.64	76.42
SAU	**71.62**	**77.47**

4.2 Object Detection

A standard problem in computer vision is object detection, in which the network model tries to locate and identify each object present in the image. Object detection is widely used in face detection, image retrieval, autonomous vehicle etc. In this section, we present our results on challenging Pascal VOC dataset ([5]) on Single Shot MultiBox Detector(SSD) 300 ([25]) and we consider VGG-16(with batch-normalization) ([40]) model as the backbone network. No pre-trained weight is considered for our experiments in the network. The network has been trained with a batch size of 8, SGD optimizer ([37], [17]) with 0.9 momentum, $5e^{-4}$ weight decay, 0.001 learning rate, and trained up to 120000 iterations. We report the mean average precision (mAP) in Table 9 for the mean of 6 different runs.

Table 9: A Detailed Comparison between SAU Activation Function and Other Baseline Activation Functions on the Pascal VOC Dataset for Object Detection Problem with SSD300 Network Architecture. mAP is Reported for the Mean of 6 Different Runs in the Table. We Report mean±std in the Table.

Activation Function	mAP
ReLU	77.2±0.14
Leaky ReLU	77.2±0.19
PReLU	77.2±0.20
ReLU6	77.1±0.15
ELU	75.1±0.22
Softplus	74.2±0.25
PAU	77.4±0.14
Swish	77.3±0.11
GELU	77.3±0.12
SAU	**77.7**±0.10

4.3 Semantic Segmentation

Semantic segmentation is a computer vision problem that narrates the procedure of associating each pixel of an image with a class label. We present our experimental results in this section on the popular Cityscapes dataset ([4]). The U-net model ([38]) is considered as the segmentation framework and is trained up-to 250 epochs, with adam optimizer ([18]), learning rate $5e^{-3}$, and batch size 32. We report the mean of 6 different runs for pixel accuracy and the mean Intersection-Over-Union (mIOU) on test data in table 10.

Table 10: A Detailed Comparison between SAU Activation Function and Other Baseline Activation Functions on the CityScapes Dataset for Semantic Segmentation Problem on U-Net Model. Pixel Accuracy and mIOU is Reported for the Mean of 6 Different Runs in the Table. We report mean±std in the Table.

Activation Function	Pixel Accuracy	mIOU
ReLU	79.45±0.47	69.39±0.28
PReLU	78.88±0.40	68.80±0.40
ReLU6	79.67±0.40	69.79±0.42
Leaky ReLU	79.32±0.40	69.60±0.40
ELU	79.38±0.51	68.10±0.40
Softplus	78.60±0.49	68.20±0.49
PAU	79.52±0.49	69.12±0.31
Swish	79.99±0.47	69.61±0.29
GELU	80.10±0.37	69.39±0.38
SAU	**81.11**±0.40	**71.02**±0.32

4.4 Machine Translation

Machine Translation is a deep learning technique in which a model translates text or speech from one language to another language. In this section, we report results on WMT 2014, English→German dataset. The database has 4.5 million training sentences. Network performance is evaluated on the newstest2014 dataset using the BLEU score metric. An Attention-based 8-head transformer network ([44]) in trained with Adam optimizer ([18]), 0.1 dropout rate ([41]), and trained up to 100000 steps. Other hyperparameters are kept similar, as mentioned in the original paper ([44]). We report the mean of 6 different runs on Table 11 on the test dataset(newstest2014).

5 Baseline Table

In this section, we present a table for SAU and the other baseline functions, which shows that SAU beat or performs equally well compared to baseline activation functions in most cases. We report a detailed comparison with SAU and the baseline activation functions based on all the experiments in earlier sections and supplementary material in Table 12. We notice that SAU performs remarkably well in most of the cases when compared with the baseline activations.

6 Conclusion

In this paper, we propose a new novel smooth activation function using approximate identity, and we call it a smooth activation unit (SAU). The proposed

Table 11: A Detailed Comparison between SAU Activation Function and Other Baseline Activation Functions on the WMT-2014 Dataset for Machine Translation Problem on Transformer Model. BLEU Score is Reported for the Mean of 6 Different Runs in the Table. We Report mean±std in the Table.

Activation Function	BLEU Score on the newstest2014 dataset
ReLU	26.2±0.15
Leaky ReLU	26.3±0.17
PReLU	26.2±0.21
ReLU6	26.1±0.14
ELU	25.1±0.15
Softplus	23.6±0.16
PAU	26.3±0.14
Swish	26.4±0.10
GELU	26.4±0.19
SAU	**26.7**±0.12

Table 12: Baseline Table for SAU. In this Table, We Report the Total Number of Cases in Which SAU Underperforms, Equal, or Outperforms When We Compare it with the Baseline Activation Functions

Baselines	ReLU	Leaky ReLU	PReLU	ReLU6	ELU	Softplus	PAU	Swish	GELU
SAU > Baseline	71	71	71	71	71	72	67	66	67
SAU = Baseline	0	0	0	0	0	0	0	0	0
SAU < Baseline	1	1	1	1	1	0	5	6	5

function can approximate ReLU or its different variants (like Leaky ReLU etc.) quite well. For our experiments, we consider SAU as a trainable activation function, and we show that in a wide range of experiments on different deep learning problems, the proposed functions outperform the known activations like ReLU, Leaky ReLU or Swish in most cases which shows that replacing the hand-crafted activation functions by SAU can be beneficial in deep networks.

Acknowledgement: The authors are very grateful to Dr. Bapi Chatterjee for lending GPU equipment which helped in carrying out some of the important experiments.

References

1. Chollet, F., et al.: Keras. https://keras.io (2015)
2. Chollet, F.: Xception: Deep learning with depthwise separable convolutions (2017)
3. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus) (2016)
4. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding (2016)
5. Everingham, M., Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. Int. J. Comput. Vision **88**(2), 303–338 (Jun 2010). https://doi.org/10.1007/s11263-009-0275-4, https://doi.org/10.1007/s11263-009-0275-4
6. Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout networks (2013)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
8. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification (2015)
9. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks (2016)
10. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus) (2020)
11. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017)
12. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7132–7141 (2018). https://doi.org/10.1109/CVPR.2018.00745
13. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks (2016)
14. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size (2016)
15. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift (2015)
16. Kidger, P., Lyons, T.: Universal approximation with deep narrow networks (2020)
17. Kiefer, J., Wolfowitz, J.: Stochastic estimation of the maximum of a regression function. Annals of Mathematical Statistics **23**, 462–466 (1952)
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1412.6980
19. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
20. Krizhevsky, A.: Convolutional deep belief networks on cifar-10 (2010)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. p. 1097–1105. NIPS'12, Curran Associates Inc., Red Hook, NY, USA (2012)

22. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4), 541–551 (1989). https://doi.org/10.1162/neco.1989.1.4.541
23. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
24. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist **2** (2010)
25. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. Lecture Notes in Computer Science p. 21–37 (2016). https://doi.org/10.1007/978-3-319-46448-0_2, `http://dx.doi.org/10.1007/978-3-319-46448-0_2`
26. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts (2017)
27. Ma, N., Zhang, X., Liu, M., Sun, J.: Activate or not: Learning customized activation (2021)
28. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design (2018)
29. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: in ICML Workshop on Deep Learning for Audio, Speech and Language Processing (2013)
30. Misra, D.: Mish: A self regularized non-monotonic activation function (2020)
31. Molina, A., Schramowski, P., Kersting, K.: Padé activation units: End-to-end learning of flexible activation functions in deep networks (2020)
32. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel. pp. 807–814. Omnipress (2010), `https://icml.cc/Conferences/2010/papers/432.pdf`
33. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
34. Nickolls, J., Buck, I., Garland, M., Skadron, K.: Scalable parallel programming. In: 2008 IEEE Hot Chips 20 Symposium (HCS). pp. 40–53 (2008). https://doi.org/10.1109/HOTCHIPS.2008.7476525
35. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library (2019)
36. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions (2017)
37. Robbins, H., Monro, S.: A stochastic approximation method. Annals of Mathematical Statistics **22**, 400–407 (1951)
38. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation (2015)
39. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks (2019)
40. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015)
41. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (Jan 2014)

42. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision (2015)
43. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks (2020)
44. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2017)
45. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
46. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks (2017)
47. Zagoruyko, S., Komodakis, N.: Wide residual networks (2016)
48. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=r1Ddp1-Rb
49. Hui-zhen Zhao, Fu-xian Liu, L.y.L.: Improving deep convolutional neural networks with mixed maxout units (2017). https://doi.org/https://doi.org/10.1371/journal.pone.0180049
50. Zheng, H., Yang, Z., Liu, W., Liang, J., Li, Y.: Improving deep neural networks using softplus units. In: 2015 International Joint Conference on Neural Networks (IJCNN). pp. 1–4 (2015). https://doi.org/10.1109/IJCNN.2015.7280459