SAU: SMOOTH ACTIVATION FUNCTION USING CONVOLUTION WITH APPROXIMATE IDENTITIES

Anonymous authors
Paper under double-blind review

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 5.12% improvement with ShuffleNet V2 (2.0x) model on the CIFAR100 dataset.

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) (Nair & Hinton (2010)), Leaky ReLU (Maas et al. (2013)) 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 (Clevert et al. (2016)), Parametric ReLU (PReLU) (He et al. (2015a)), Swish (Ramachandran et al. (2017)), Tanhsoft (Biswas et al. (2021d)), EIS (Biswas et al. (2021b)), Padé Activation Unit (PAU) (Molina et al. (2020)), Orthogonal Padé Activation Unit (OPAU) (Biswas et al. (2021a)), ACON (Ma et al. (2021)), ErfAct (Biswas et al. (2021c)), Mish (Misra (2020)), GELU (Hendrycks & Gimpel (2020)), ReLU6 (Krizhevsky (2010)), Softplus (Zheng et al. (2015)) 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 (Ramachandran et al. (2017)) 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 (He et al. (2015a)) tries to overcome this problem by introducing a learnable negative component to ReLU (Nair & Hinton (2010)). Maxout (Goodfellow et al. (2013)) and Mixout (Hui-zhen Zhao (2017)) 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 (Ma et al. (2021)), a smooth activation, have 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 (Molina et al. (2020)) and OPAU (Biswas et al. (2021a)) are two promising candidates 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 as 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 larter 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 equation 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. $\lim\limits_{\epsilon \to 0} \phi_\epsilon(x) := \lim\limits_{\epsilon \to 0} \dfrac{1}{\epsilon}\phi(x/\epsilon) = \delta(x)$, 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 equation 2.

3 SMOOTH APPROXIMATION UNIT (SAU)

Consider the Gaussian function

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

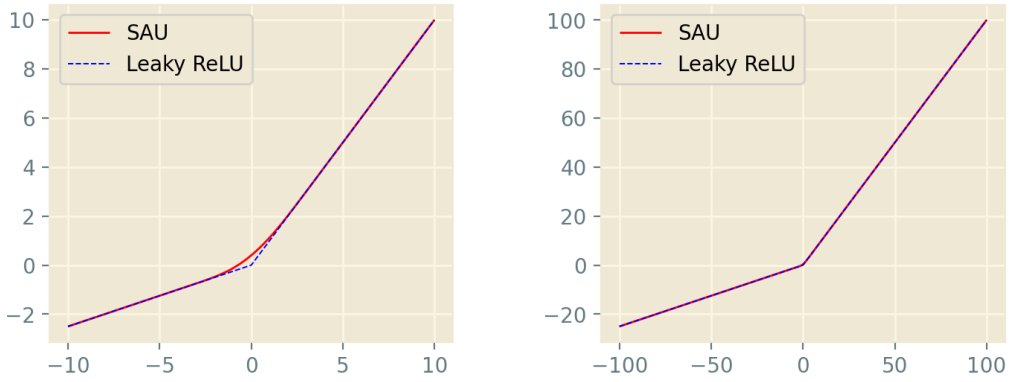

Figure 1: Approximation of Leaky ReLU ($\alpha = 0.25$) using SAU. The left figure shows that SAU approximate Leaky ReLU smoothly, and in the right figure, we plot the same functions on a larger domain range.

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

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

Note that $\text{LeakyReLU}[\alpha]$ activation function is hyperparametrized by α and it is non-differentiable at the origin for all values of α except $\alpha = 1$. For $\alpha = 0$, $\text{LeakyReLU}[\alpha]$ reduces to well known activation function ReLU (Nair & Hinton (2010)) while for constant and trainable α, $\text{LeakyReLU}[\alpha]$ reduces to Leaky ReLU (Maas et al. (2013)) and Parametric ReLU (He et al. (2015a)) respectively. For a given $n \in \mathbb{N}$, and $\alpha \neq 1$, a smooth approximation of $\text{LeakyReLU}[\alpha]$ 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

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

It is noticeable that this function is not zero centered but passes by extremely close neighbourhood of zero. To make the function zero centered, we have multiplied the first term of (3) by a linear component x. To further investigate these two functions as a possible candidates for activation function, we have conducted several experiments on MNIST (LeCun et al. (2010)), CIFAR10 (Krizhevsky (2009)), and CIFAR100 (Krizhevsky (2009)) datasets with PreActResNet-18 (He et al. (2016)), VGG-16 (with batch-normalization) (Ioffe & Szegedy (2015)) (Simonyan & Zisserman (2015)), and DenseNet-121 (Huang et al. (2016)) models, and we notice that both of them performs almost similar in every cases. So, for rest of the paper, we will only consider the approximate identity of Leaky ReLU ($\alpha = 0.25$) given in (3) as the activation function. We call this function Smooth Approximation 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 approximate Leaky ReLU (as well as ReLU or its variants) quite well.

We note that in GELU (Hendrycks & Gimpel (2020)) paper, a similar idea is utilized to obtain their activation functions. They use the product of x with the cumulative distribution function of a suitable probability distribution (see (Hendrycks & Gimpel (2020)) for further details).

3.1 Learning activation parameters via back-propagation

Back-propagation algorithm (LeCun et al. (1989)) 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 (Paszke et al. (2019)) & Tensorflow-Keras (Chollet et al. (2015)) API, and automatic differentiation will update the parameters. Alternatively, CUDA (Nickolls et al. (2008)) based implementation (see (Maas et al. (2013))) can be used and the gradients of equation (3) for the input x and the parameter α can be computed as follows:

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

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

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 (Molina et al. (2020))).

Proposition 1. (Theorem 1.1 in Kidger and Lyons, 2020 (Kidger & Lyons (2020))) :- 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 next sections that SAU outperform in most cases compared to the standard activations. For the rest of our experiments, we have considered α as a trainable parameter and n as a hyperparameter. We have initialized α at 0.25 and updated it via backpropagation according to (5). The value of n is considered 20000. All the experiments are conducted on an NVIDIA V100 GPU with 32GB RAM.

4.1 Image Classification

4.1.1 MNIST, Fashion MNIST and The Street View House Numbers (SVHN) Database

In this section, we present results on MNIST (LeCun et al. (2010)), Fashion MNIST (Xiao et al. (2017)), and SVHN (Netzer et al. (2011)) 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, shearing on the three datasets. We report results with AlexNet (Krizhevsky et al. (2012)) architecture in Table 1. For all the experiments to train a model on these three datasets, we use a batch size of 128, stochastic gradient descent (Robbins & Monro (1951), Kiefer & Wolfowitz (1952)) 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 cosine annealing (Loshchilov & Hutter (2017)) learning rate scheduler. We report more experiments on these datasets with LeNet (Lecun et al. (1998)), VGG-16 (Simonyan & Zisserman (2015)) (with batch-normalization (Ioffe & Szegedy (2015))) and a custom-designed model in the appendix section.

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

Table 1: A detailed comparison between SAU activation and other baseline activations in MNIST, Fashion MNIST, and SVHN datasets for image classification problem on AlexNet architecture. We report top-1 test accuracy (in %) for the mean of 10 different runs. mean$\pm$std is reported in the table.

4.1.2 CIFAR

The CIFAR (Krizhevsky (2009)) is one of the most popular databases for image classification 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 2 and Table 3 on CIFAR10 and CIFAR100 datasets respectively. We consider MobileNet V1 (Howard et al. (2017)), MobileNet V2 (Sandler et al. (2019)), Shufflenet V2 (Ma et al. (2018)), PreActResNet (He et al. (2016)), ResNet (He et al. (2015b)), Inception V3 (Szegedy et al. (2015)), DenseNet (Huang et al. (2016)), WideResNet (Zagoruyko & Komodakis (2016)), VGG (Simonyan & Zisserman (2015)) (with batch-normalization (Ioffe & Szegedy (2015))), and EfficientNet B0 (Tan & Le (2020)). For all the experiments to train a model on these two datasets, we use a batch size of 128, stochastic gradient descent (Robbins & Monro (1951), Kiefer & Wolfowitz (1952)) 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 with cosine annealing (Loshchilov & Hutter (2017)) learning rate scheduler. Standard data augmentation methods like width shift, height shift, horizontal flip, and rotation is applied on both datasets. It is noticeable from these two tables that replacing ReLU by SAU, there is an increment in top-1 accuracy from 1% to more than 5% in most of the models. A more detailed result on these two datasets with the other baseline activations are reported in the appendix section. 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 few epochs, SAU has stable & smooth learning, faster convergence speed, and higher accuracy and lower loss on the test dataset compared to other baseline activation functions.

4.1.3 TINY IMAGENET

In this section, we present results on the Tiny ImageNet dataset, a similar kind of image classification database like the ImageNet Large Scale Visual Recognition Challenge(ILSVRC). Tiny Imagenet 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 4 on WideResNet 28-10 (WRN 28-10) (Zagoruyko & Komodakis (2016)) model. We consider a batch size of 64, 0.2 dropout rate (Srivastava et al. (2014)), SGD optimizer (Robbins & Monro (1951), Kiefer & Wolfowitz (1952)), He Normal initializer (He et al. (2015a)), 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, fill mode is 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$\pm$std) and got a good improvement for SAU over ReLU.

Model	ReLU	SAU
	Top-1 accuracy (mean± std)	Top-1 accuracy (mean ± std)
ShuffleNet V2 0.5x	88.28 ± 0.21	**90.50** ± 0.17
ShuffleNet V2 1.0x	90.96 ± 0.23	**92.78** ± 0.20
ShuffleNet V2 1.5x	91.29 ± 0.19	**93.20** ± 0.18
ShuffleNet V2 2.0x	91.79 ± 0.20	**93.52** ± 0.16
PreActResNet 18	93.63 ± 0.16	**94.92** ± 0.14
PreActResNet 34	94.12 ± 0.17	**95.17** ± 0.14
PreActResNet 50	94.20 ± 0.16	**95.14** ± 0.15
ResNet 18	93.51 ± 0.18	93.39 ± 0.17
ResNet 34	93.97 ± 0.20	93.88 ± 0.16
MobileNet V1	92.49 ± 0.20	**93.54** ± 0.14
MobileNet V2	94.14 ± 0.17	**95.37** ± 0.09
Inception V3	94.15 ± 0.14	**94.84** ± 0.09
WideResNet 28-10	95.10 ± 0.16	**95.94** ± 0.11
DenseNet 121	94.72 ± 0.15	**95.70** ± 0.09
EffitientNet B0	95.12 ± 0.15	**96.21** ± 0.09
VGG16	93.56 ± 0.21	93.07 ± 0.20

Table 2: A detailed comparison between SAU activation and other baseline activations in the CIFAR10 dataset for image classification problem on different popular network architectures. We report top-1 test accuracy (in %) for the mean of 10 different runs. mean±std is reported in the table.

Model	ReLU	SAU
	Top-1 accuracy (mean± std)	Top-1 accuracy (mean ± std)
Shufflenet V2 0.5x	62.15 ± 0.25	**64.84** ± 0.22
Shufflenet V2 1.0x	64.50 ± 0.26	**68.79** ± 0.23
Shufflenet V2 1.5x	67.42 ± 0.30	**72.69** ± 0.27
Shufflenet V2 2.0x	67.91 ± 0.27	**73.03** ± 0.22
PreActResNet 18	72.22 ± 0.25	**74.51** ± 0.22
PreActResNet 34	73.32 ± 0.26	**75.79** ± 0.24
PreActResNet 50	73.49 ± 0.28	**75.82** ± 0.25
ResNet 18	73.26 ± 0.26	**74.47** ± 0.25
ResNet 34	73.41 ± 0.27	74.91 ± 0.22
MobileNet V1	71.22 ± 0.24	**72.36** ± 0.24
MobileNet V2	74.10 ± 0.24	**75.87** ± 0.20
Inception V3	74.25 ± 0.26	**76.21** ± 0.21
WideResNet 28-10	76.32 ± 0.24	**77.65** ± 0.19
DenseNet 121	75.99 ± 0.27	**77.23** ± 0.24
EffitientNet B0	76.53 ± 0.26	**78.27** ± 0.25
VGG16	71.91 ± 0.30	71.31 ± 0.29

Table 3: A detailed comparison between SAU activation and other baseline activations in the CIFAR100 dataset for image classification problem on different popular network architectures. We report top-1 test accuracy (in %) for the mean of 10 different runs. mean±std is reported in the table.

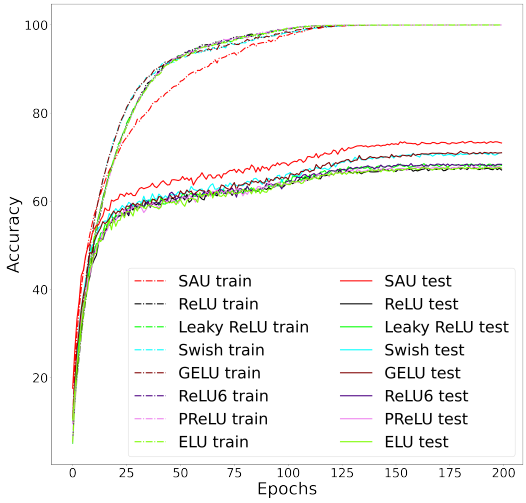

Figure 2: Top-1 train and test accuracy curves (higher is better) for SAU and baseline activation functions on CIFAR100 dataset with ShuffleNet V2 (2.0x) model.

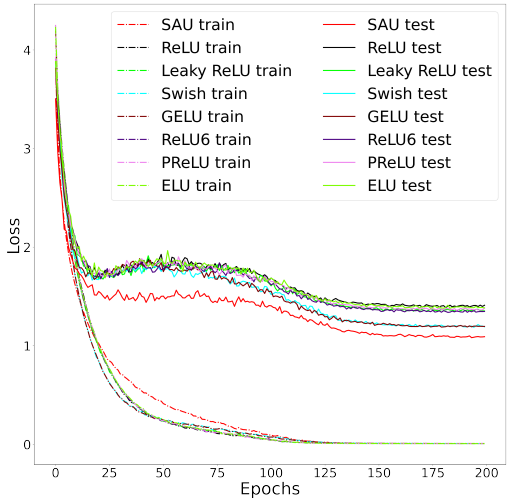

Figure 3: Top-1 train and test loss curves (lower is better) for SAU and baseline activation functions on CIFAR100 dataset with ShuffleNet V2 (2.0x) model.

Activation Function	top-1 Accuracy
SAU	**64.07**±0.44
ReLU	62.77±0.46
Leaky ReLU	62.72±0.46
PReLU	62.70±0.48
ReLU6	62.59±0.46
ELU	62.58±0.50
Softplus	61.77±0.59
PAU	63.62±0.44
Swish	63.47±0.46
GELU	63.26±0.48

Table 4: A detailed comparison between SAU activation and other baseline activations in Tiny ImageNet dataset for image classification problem on WideResNet 28-10 architecture. We report top-1 test accuracy (in %) for the mean of 6 different runs. mean±std is reported in the table.

4.2 Object Detection

A standard problem in computer vision is object detection, in which the network model try to locate and identify each object present in the image. Object detection is widely used in face detection, autonomous vehicle etc. In this section, we present our results on challenging Pascal VOC dataset (Everingham et al. (2010)) on Single Shot MultiBox Detector(SSD) 300 (Liu et al. (2016)) with VGG-16(with batch-normalization) (Simonyan & Zisserman (2015)) 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 (Robbins & Monro (1951); Kiefer & Wolfowitz (1952)) 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 5 for the mean of 6 different runs.

Activation Function	mAP
SAU	**77.7**±0.10
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

Table 5: A detailed comparison between SAU activation and other baseline activations in Pascal VOC dataset for object detection problem on SSD300 network architecture. We report mAP for the mean of 6 different runs. mean±std is reported in the table.

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 (Cordts et al. (2016)). The U-net model (Ronneberger et al. (2015)) is considered as the segmentation framework and is trained up-to 250 epochs, with adam optimizer (Kingma & Ba (2015)), learning rate $5e^{-3}$, batch size 32 and Xavier Uniform initializer (Glorot & Bengio (2010)). We report the mean of 6 different runs for Pixel Accuracy and the mean Intersection-Over-Union (mIOU) on test data on table 6.

Activation Function	Pixel Accuracy	mIOU
SAU	**81.11**±0.40	**71.02**±0.32
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

Table 6: A detailed comparison between SAU activation and other baseline activations in CityScapes dataset for semantic segmentation problem on U-NET model. We report pixel accuracy and mIOU for the mean of 6 different runs. mean±std is reported in the table.

4.4 MACHINE TRANSLATION

Machine Translation is a deep learning technique in which a model translate text or speech from one language to another language. In this section, we report results on WMT 2014 English→German dataset. The database contains 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 (Vaswani et al. (2017)) in trained with Adam optimizer (Kingma & Ba (2015)), 0.1 dropout rate (Srivastava et al. (2014)), and trained up to 100000 steps. Other hyperparameters are kept similar as mentioned in the original paper (Vaswani et al. (2017)). We report the mean of 6 different runs on Table 7 on the test dataset(newstest2014).

Activation Function	BLEU Score on the newstest2014 dataset
SAU	**26.7**±0.12
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

Table 7: A detailed comparison between SAU activation and other baseline activations in WMT-2014 dataset for machine translation problem on transformer model. We report BLEU score for the mean of 6 different runs. mean±std is reported in the table.

5 BASELINE TABLE

In this section, we present a table for SAU and the other baseline functions, which shows that SAU beat or perform 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 in Table 8. We notice that SAU performs remarkably well in most of the cases when compared with the baseline activations.

Baselines	ReLU	Leaky ReLU	PReLU	ReLU6	ELU	Softplus	PAU	Swish	GELU
SAU $>$ Baseline	43	43	43	43	43	48	41	41	43
SAU $=$ Baseline	0	0	0	0	0	0	0	1	0
SAU $<$ Baseline	5	5	5	5	5	0	7	6	5

Table 8: Baseline table for SAU. In the table, we report the total number of cases in which SAU underperforms, equal, or outperforms when we compare with the baseline activation functions

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 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 on 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.

REFERENCES

Koushik Biswas, Shilpak Banerjee, and Ashish Kumar Pandey. Orthogonal-padé activation functions: Trainable activation functions for smooth and faster convergence in deep networks, 2021a.

Koushik Biswas, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. Eis - efficient and trainable activation functions for better accuracy and performance. In Igor Farkaš, Paolo Masulli, Sebastian Otte, and Stefan Wermter (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2021*, pp. 260–272, Cham, 2021b. Springer International Publishing. ISBN 978-3-030-86340-1.

Koushik Biswas, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. Erfact and pserf: Non-monotonic smooth trainable activation functions, 2021c.

Koushik Biswas, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. Tanhsoft—dynamic trainable activation functions for faster learning and better performance. *IEEE Access*, 9:120613–120623, 2021d. doi: 10.1109/ACCESS.2021.3105355.

François Chollet et al. Keras. `https://keras.io`, 2015.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010. ISSN 0920-5691. doi: 10.1007/s11263-009-0275-4. URL `https://doi.org/10.1007/s11263-009-0275-4`.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. JMLR Workshop and Conference Proceedings. URL `http://proceedings.mlr.press/v9/glorot10a.html`.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2020.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016.

Long-yue Li Hui-zhen Zhao, Fu-xian Liu. Improving deep convolutional neural networks with mixed maxout units, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks, 2020.

J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky. Convolutional deep belief networks on cifar-10, 2010.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, pp. 21–37, 2016. ISSN 1611-3349. doi: 10.1007/978-3-319-46448-0_2. URL http://dx.doi.org/10.1007/978-3-319-46448-0_2.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design, 2018.

Ningning Ma, Xiangyu Zhang, Ming Liu, and Jian Sun. Activate or not: Learning customized activation, 2021.

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

Diganta Misra. Mish: A self regularized non-monotonic activation function, 2020.

Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks, 2020.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pp. 807–814. Omnipress, 2010. URL https://icml.cc/Conferences/2010/papers/432.pdf.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming. In *2008 IEEE Hot Chips 20 Symposium (HCS)*, pp. 40–53, 2008. doi: 10.1109/HOTCHIPS.2008.7476525.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, January 2014. ISSN 1532-4435.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016.

Hao Zheng, Zhanlei Yang, Wenju Liu, Jizhong Liang, and Yanpeng Li. Improving deep neural networks using softplus units. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–4, 2015. doi: 10.1109/IJCNN.2015.7280459.

A APPENDIX

A.1 BASELINE ACTIVATION FUNCTIONS

1. **ReLU:** ReLU (Nair & Hinton (2010)) is a widely used activation function and defined as $f(x) = max(x, 0)$.
2. **ELU:** ELU (Clevert et al. (2016)) is defined as $f(x) = max(x, 0) + min(\alpha(e^x - 1), 0)$.
3. **Softplus:** Softplus (Zheng et al. (2015)) is a smoothing approximation of ReLU and defined as $f(x) = ln(1 + e^x)$.
4. **Leaky ReLU:** Leaky ReLU (Maas et al. (2013)) is a variant of ReLU and is defined as $f(x) = max(x, 0.01x)$
5. **ReLU6:** ReLU6 (Krizhevsky (2010)) is a bounded variant of ReLU defined as $f(x) = min(max(x, 0), 6)$.
6. **Parametric ReLU:** Parametric ReLU (PReLU) (He et al. (2015a)) is a trainable activation function and is defined as $f(x) = max(x, \alpha x)$ where α is the learnable parameter.
7. **PAU:** PAU (Molina et al. (2020)) is a smooth approximation of Leaky ReLU function by rational approximation of polynomials up-to a given order. PAU is a trainable activation function.
8. **Swish:** Swish (Ramachandran et al. (2017)) is a popular activation function and defined as $f(x) = \dfrac{x}{1 + e^{-\beta x}}$, where β is either a hyperparameter or trainable parameter.
9. **GELU:** GELU (Hendrycks & Gimpel (2020)) can be viewed as a smooth approximation of ReLU and GELU is defined as $f(x) = \frac{x}{2}(1 + \text{erf}(\frac{x}{\sqrt{2}}))$.

A.2 MNIST, FASHION MNIST, AND SVHN

We report a more detailed experiment on MNIST, Fashion MNIST, and SVHN datasets on VGG-16, LeNet, and a custom-designed model in Table 9, 10, and 11 respectively. We design the custom network with CNN layers with 3×3 kernels and max-pooling layers with 2×2 kernels. We consider Channel depths of size 128 (twice), 64 (thrice), 32 (twice), with a dense layer of size 128, Max-pooling layer(thrice), and dropout (Srivastava et al. (2014)). We have applied batch-normalization (Ioffe & Szegedy (2015)) before the activation function layer, and more details about this architecture are given in figure 4. For the experimental setup on these three networks, we follow the same experimental setup we use for AlexNet architecture on these three datasets. The results reported in Table 9, 10, and 11 is also counted in the baseline table. Best performing activation(s) is(are) reported in bold text.

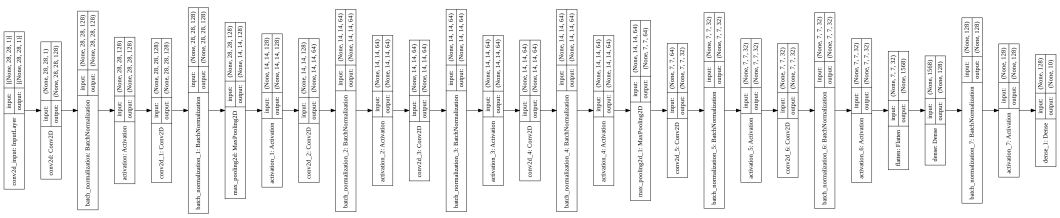

Figure 4: Custom designed network used for evaluation on MNIST, Fashion MNIST, and SVHN

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

Table 9: A detailed comparison between SAU activation and other baseline activations in MNIST, Fashion MNIST, and SVHN datasets for image classification problem on VGG16 architecture. We report top-1 test accuracy (in %) for the mean of 10 different runs. mean$\pm$std is reported in the table.

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

Table 10: A detailed comparison between SAU activation and other baseline activations in MNIST, Fashion MNIST, and SVHN datasets for image classification problem on LeNet architecture. We report top-1 test accuracy (in %) for the mean of 10 different runs. mean$\pm$std is reported in the table.

Activation Function	MNIST	Fashion MNIST	SVHN
SAU	99.57 $\pm$ 0.04	**93.50** $\pm$ 0.09	**95.10** $\pm$ 0.09
ReLU	99.48 $\pm$ 0.07	93.06 $\pm$ 0.14	94.52 $\pm$ 0.13
Leaky ReLU	99.43 $\pm$ 0.08	93.22 $\pm$ 0.15	94.67 $\pm$ 0.16
PReLU	99.41 $\pm$ 0.09	93.14 $\pm$ 0.12	94.61 $\pm$ 0.15
ReLU6	99.45 $\pm$ 0.07	93.31 $\pm$ 0.14	94.70 $\pm$ 0.10
ELU	99.50 $\pm$ 0.05	93.28 $\pm$ 0.16	94.62 $\pm$ 0.13
Softplus	99.32 $\pm$ 0.10	92.95 $\pm$ 0.20	94.41 $\pm$ 0.18
PAU	**99.58**$\pm$0.09	93.26 $\pm$ 0.16	94.89 $\pm$ 0.11
Swish	99.57$\pm$0.08	93.29 $\pm$ 0.14	94.72 $\pm$ 0.12
GELU	99.50$\pm$ 0.05	93.35 $\pm$ 0.12	94.79 $\pm$ 0.09

Table 11: A detailed comparison between SAU activation and other baseline activations in MNIST, Fashion MNIST, and SVHN datasets for image classification problem on Custom-designated architecture. We report top-1 test accuracy (in %) for the mean of 10 different runs. mean$\pm$std is reported in the table.

A.3 Experimental results with baselines on CIFAR datasets

We report a more detailed experiments on CIFAR10 and CIFAR 100 datasets with Leaky ReLU, ReLU6, PReLU, ELU, Softplus, Swish, and GELU activations. Experimental results on CIFAR10 are reported in Table 12, 13 and experimental results on CIFAR100 are reported in Table 14 and

15. The results are reported on MobileNet V1 (MN V1) (Howard et al. (2017)), MobileNet V2 (MN V2) (Sandler et al. (2019)),) Shufflenet V2 (SF V2) (Ma et al. (2018)), PreActResNet (PARN) (He et al. (2016)), ResNet (RN) (He et al. (2015b)), Inception V3 (IN V3) (Szegedy et al. (2015)), DenseNet (DN) (Huang et al. (2016)), WideResNet (WRN) (Zagoruyko & Komodakis (2016)), VGG (Simonyan & Zisserman (2015)) (with batch-normalization (Ioffe & Szegedy (2015))), and EfficientNet B0 (EN B0) (Tan & Le (2020)) networks architectures. We follow the same hyper-parameter setup as described in section 4.1.2. To count the numbers in the baseline table for all baseline activations, we also consider the results reported in Table 12, 13, 14, 15 along with results reported in Table 2 and 3. Comparing these six tables it is clear that SAU beats all the baseline activations in most of the cases on CIFAR10 and CIFAR100 datasets.

Activation Function	SF V2 0.5x	SF V2 1.0x	SF V2 1.5x	SF V2 2.0x	MN V1	MN V2	RN 18	RN 34
Leaky ReLU	88.48 ±0.22	91.10 ±0.24	91.36 ±0.21	91.65 ±0.22	92.62 ±0.22	94.21 ±0.17	93.57 ±0.20	93.85 ±0.19
PReLU	88.10 ±0.24	91.17 ±0.25	91.21 ±0.26	91.85 ±0.21	92.51 ±0.24	94.37 ±0.19	93.52 ±0.20	93.65 ±0.21
ReLU6	88.45 ±0.20	91.29 ±0.23	91.52 ±0.20	91.71 ±0.20	92.70 ±0.21	94.17 ±0.19	93.67 ±0.23	93.99 ±0.22
ELU	89.17 ±0.21	90.90 ±0.25	91.49 ±0.20	91.77 ±0.23	92.77 ±0.20	94.01 ±0.16	93.71 ±0.18	93.89 ±0.20
Softplus	88.89 ±0.32	90.77 ±0.30	90.97 ±0.29	91.10 ±0.28	92.13 ±0.31	94.10 ±0.24	93.59 ±0.29	93.55 ±0.28
PAU	89.49 ±0.22	92.10 ±0.27	92.49 ±0.16	92.62 ±0.19	93.21 ±0.15	94.89 ±0.14	93.72 ±0.20	94.01 ±0.16
Swish	89.01 ±0.20	91.67 ±0.25	91.89 ±0.19	92.32 ±0.18	93.12 ±0.17	94.61 ±0.12	93.67 ±0.18	94.10 ±0.16
GELU	89.10 ±0.20	91.45 ±0.26	91.69 ±0.20	92.37 ±0.18	92.97 ±0.19	94.73 ±0.13	93.66 ±0.20	94.17 ±0.15

Table 12: Experimental results for baseline activations in CIFAR100 dataset for image classification problem on different popular network architectures. We report top-1 test accuracy (in %) for the mean of 10 different runs. mean±std is reported in the table.

Activation Function	PARN 18	PARN 34	PARN 50	IN V3	WRN 28-10	DN 121	EN B0	VGG 16
Leaky ReLU	93.50 ±0.16	94.20 ±0.18	94.35 ±0.17	94.35 ±0.15	95.05 ±0.16	94.79 ±0.15	95.21 ±0.17	93.65 ±0.22
PReLU	93.57 ±0.20	94.27 ±0.22	94.30 ±0.16	94.42 ±0.19	95.01 ±0.17	94.65 ±0.12	95.29 ±0.19	93.50 ±0.25
ReLU6	93.65 ±0.16	94.23 ±0.19	94.47 ±0.20	94.27 ±0.22	95.35 ±0.22	94.68 ±0.29	95.37 ±0.16	93.66 ±0.22
ELU	93.79 ±0.15	94.55 ±0.20	94.30 ±0.16	94.52 ±0.17	95.17 ±0.19	94.66 ±0.18	95.09 ±0.12	93.50 ±0.23
Softplus	93.10 ±0.22	93.98 ±0.19	94.39 ±0.18	94.09 ±0.20	94.97 ±0.16	94.71 ±0.20	95.10 ±0.20	93.01 ±0.27
PAU	94.41 ±0.14	94.71 ±0.17	94.79 ±0.16	94.57 ±0.14	95.48 ±0.15	94.99 ±0.14	95.51 ±0.16	93.62 ±0.21
Swish	94.12 ±0.15	94.54 ±0.15	94.34 ±0.14	94.45 ±0.13	95.39 ±0.16	94.82 ±0.13	95.37 ±0.14	93.71 ±0.20
GELU	94.19 ±0.15	94.45 ±0.16	94.45 ±0.14	94.39 ±0.14	95.49 ±0.15	94.89 ±0.14	95.17 ±0.14	93.82 ±0.18

Table 13: Experimental results for baseline activations in CIFAR10 dataset for image classification problem on different popular network architectures. We report top-1 test accuracy (in %) for the mean of 10 different runs. mean±std is reported in the table.

Activation Function	SF V2 0.5x	SF V2 1.0x	SF V2 1.5x	SF V2 2.0x	MN V1	MN V2	RN 18	RN 34
Leaky ReLU	62.12 ±0.32	65.03 ±0.30	67.23 ±0.29	68.10 ±0.29	71.39 ±0.32	74.08 ±0.27	73.56 ±0.30	74.58 ±0.27
PReLU	61.89 ±0.35	64.83 ±0.32	67.16 ±0.34	68.45 ±0.31	71.56 ±0.33	74.38 ±0.29	73.79 ±0.32	74.65 ±0.28
ReLU6	62.32 ±0.29	64.69 ±0.30	67.87 ±0.27	68.65 ±0.27	71.58 ±0.30	74.04 ±0.28	73.64 ±0.30	74.81 ±0.28
ELU	62.36 ±0.29	65.50 ±0.27	67.89 ±0.28	68.39 ±0.30	71.25 ±0.30	74.18 ±0.28	73.69 ±0.29	74.77 ±0.29
Softplus	61.99 ±0.35	64.46 ±0.31	67.01 ±0.33	69.01 ±0.28	71.01 ±0.39	73.56 ±0.37	73.14 ±0.34	74.18 ±0.29
PAU	63.54 ±0.27	66.84 ±0.24	70.12 ±0.24	70.89 ±0.28	71.81 ±0.24	75.31 ±0.23	73.78 ±0.24	74.99 ±0.24
Swish	63.01 ±0.25	66.07 ±0.26	69.75 ±0.26	70.12 ±0.26	71.89 ±0.25	75.17 ±0.22	73.49 ±0.26	74.96 ±0.23
GELU	63.12 ±0.26	65.87 ±0.24	70.01 ±0.28	70.23 ±0.28	71.89 ±0.26	75.24 ±0.23	73.61 ±0.29	74.76 ±0.27

Table 14: Experimental results for baseline activations in CIFAR100 dataset for image classification problem on different popular network architectures. We report top-1 test accuracy (in %) for the mean of 10 different runs. mean±std is reported in the table.

Activation Function	PARN 18	PARN 34	PARN 50	IN V3	WRN 28-10	DN 121	EN B0	VGG 16
Leaky ReLU	72.37 ±0.28	73.38 ±0.30	74.69 ±0.28	74.38 ±0.29	76.42 ±0.29	75.89 ±0.30	76.84 ±0.27	71.64 ±0.30
PReLU	72.22 ±0.30	73.51 ±0.31	74.47 ±0.29	74.32 ±0.31	76.62 ±0.30	75.95 ±0.30	76.99 ±0.29	71.81 ±0.31
ReLU6	72.51 ±0.25	73.36 ±0.31	74.92 ±0.26	74.59 ±0.26	76.82 ±0.30	76.12 ±0.28	76.76 ±0.29	71.91 ±0.27
ELU	72.54 ±0.26	73.65 ±0.29	74.72 ±0.29	74.51 ±0.28	76.12 ±0.32	75.70 ±0.31	76.82 ±0.27	71.52 ±0.29
Softplus	72.10 ±0.36	73.10 ±0.31	74.39 ±0.30	74.41 ±0.29	75.11 ±0.33	75.10 ±0.30	76.62 ±0.32	70.99 ±0.37
PAU	73.54 ±0.27	75.06 ±0.29	74.89 ±0.28	75.91 ±0.27	77.11 ±0.20	76.68 ±0.25	77.51 ±0.24	71.68 ±0.29
Swish	73.79 ±0.24	74.82 ±0.28	74.10 ±0.27	75.56 ±0.25	77.20 ±0.22	76.57 ±0.28	77.22 ±0.23	71.87 ±0.28
GELU	73.65 ±0.26	74.99 ±0.26	74.18 ±0.29	75.36 ±0.24	77.11 ±0.20	76.41 ±0.30	77.42 ±0.21	71.69 ±0.25

Table 15: Experimental results for baseline activations in CIFAR100 dataset for image classification problem on different popular network architectures. We report top-1 test accuracy (in %) for the mean of 10 different runs. mean±std is reported in the table.