Smooth Maximum Unit: Smooth Activation Function for Deep Networks using Smoothing Maximum Technique

Koushik Biswas[1], Sandeep Kumar[1, 2], Shilpak Banerjee[3], Ashish Kumar Pandey[1]
[1]IIIT Delhi [2] Shaheed Bhagat Singh College, University of Delhi [3]IIT Tirupati
{koushikb,ashish.pandey}@iiitd.ac.in, sandeep_kumar@sbs.du.ac.in, shilpak@iittp.ac.in

Abstract

Deep learning researchers have a keen interest in proposing new novel activation functions that can boost neural network performance. A good choice of activation function can have a significant effect on improving network performance and training dynamics. Rectified Linear Unit (ReLU) is a popular hand-designed activation function and is the most common choice in the deep learning community due to its simplicity though ReLU has some drawbacks. In this paper, we have proposed two new novel activation functions based on approximation of the maximum function, and we call these functions Smooth Maximum Unit (SMU and SMU-1). We show that SMU and SMU-1 can smoothly approximate ReLU, Leaky ReLU, or more general Maxout family, and GELU is a particular case of SMU. Replacing ReLU by SMU, Top-1 classification accuracy improves by 6.22%, 3.39%, 3.51%, and 3.08% on the CIFAR100 dataset with ShuffleNet V2, PreActResNet-50, ResNet-50, and SeNet-50 models respectively. Also, our experimental evaluation shows that SMU and SMU-1 improve network performance in a variety of deep learning tasks like image classification, object detection, semantic segmentation, and machine translation compared to widely used activation functions.

1. Introduction

Deep Neural network has emerged a lot in recent years and has significantly impacted our real-life applications. Neural networks are the backbone of deep learning. An activation function is the brain of the neural network, which plays a central role in the effectiveness & training dynamics of deep neural networks. Hand-designed activation functions are quite a common choice in neural network models. ReLU [36] is a widely used hand-designed activation function. Despite its simplicity, ReLU has a major drawback, known as the dying ReLU problem in which up to 50% neurons can be dead during network training. To overcome the shortcomings of ReLU, a significant number of activations have been proposed in recent years, and Leaky ReLU [33], Parametric ReLU [12], ELU [6], Softplus [56], Randomized Leaky ReLU [53] are a few of them though they marginally improve performance of ReLU. Swish [41] is a non-linear activation function proposed by the Google brain team, and it shows some good improvement of ReLU. GELU [14] is an another popular smooth activation function. It can be shown that Swish and GELU both are a smooth approximation of ReLU. Recently, a few non-linear activations have been proposed which improves the performance of ReLU, Swish or GELU. Some of them are either hand-designed or smooth approximation of Leaky ReLU function, and Mish [34], ErfAct [2], Padé activation unit [35], Orthogonal Padé activation unit [1] are a few of them.

2. Related Work and Motivation

In a deep neural network, activations are either fixed before training or trainable. Researchers have proposed several activations in recent years by combining known functions. Some of these functions have hyperparameters or trainable parameters. In the case of trainable activation functions, parameters are optimized during training. Swish is a popular activation function that can be used as either a constant or trainable activation function, and it shows some good performance in a variety of deep learning tasks like image classification, object detection, machine translation etc. GELU shares similar properties like the Swish activation function, and it gains popularity in the deep learning community due to its efficacy in natural language processing tasks. GELU has been used in BERT [8], GPT-2 [40], and GPT-3 [3] architectures. Padé activation unit (PAU) has been proposed recently, and it is constructed from the approximation of the Leaky ReLU function by rational polynomials of a given order. Though PAU improves network performance in the image classification problem over ReLU, its variants, and Swish, it has a major drawback. PAU contains many trainable parameters, which significantly increases the network complexity and computational cost.

Motivated from these works, we propose activation functions using the smoothing maximum technique. The maximum function is non-smooth at the origin. We want to explore how the smooth approximation of the maximum function (which can be used as an activation function) affects a network's training dynamics and performance. Our experimental evaluation shows that our proposed activation functions are comparatively more effective than ReLU, Mish, Swish, GELU, PAU etc., across different deep learning tasks. We summarise the paper as follows:

1. We have proposed activation functions by smoothing the maximum function. We show that it can approximate GELU, ReLU, Leaky ReLU or the general Maxout family.
2. We show that the proposed functions outperform widely used activation functions in a variety of deep learning tasks.

3. Smooth Maximum Unit

We present Smooth Maximum Unit (SMU), smooth activation functions from the smooth approximation of the maximum function. Using the smooth approximation of the $|x|$ function, one can find a general approximating formula for the maximum function, which can smoothly approximate the general Maxout [10] family, ReLU, Leaky ReLU or its variants, Swish etc. We also show that the well established GELU [14] function can be obtained as a special case of SMU.

3.1. Smooth approximation of the maximum function

Note that the maximum function can be expressed as following two different ways:

$$\begin{aligned} max(x_1, x_2) &= \begin{cases} x_1 & \text{if } x_1 \geq x_2 \\ x_2 & \text{otherwise} \end{cases} \\ &= \frac{(x_1 + x_2) + |x_1 - x_2|}{2} \end{aligned} \tag{1}$$

Note that the max function is not differentiable at the origin. Using approximations of the $|x|$ function by a smooth function, we can create approximations to the maximum functions. There are many known approximations to $|x|$, but for the rest of this article, we will focus on two specific approximations of $|x|$, namely $x\text{erf}(\mu x)$ and $\sqrt{x^2+\mu^2}$. We noticed that the activations constructed using these two functions provide good performance on standard datasets on different deep learning problems (for more details, see the supplementary section). Note that $\sqrt{x^2+\mu^2}$ as $\mu \to 0$ approximate $|x|$ from above while $x\text{erf}(\mu x)$. as $\mu \to \infty$ gives an approximation of $|x|$ from below. The approximation is uniform on compact subsets of the real line. Here erf is the Gaussian error function defined as follows:

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

Now, replacing the $|x|$ function by $x\text{erf}(\mu x)$ in equation (1), we have the smooth approximation formula for maximum function as follows:

$$f_1(x_1, x_2; \mu) = \frac{(x_1 + x_2) + (x_1 - x_2)\,\text{erf}(\mu(x_1 - x_2))}{2}. \tag{2}$$

Similarly, we can derive the the smooth approximation formula for the maximum function from equation (1) by replacing the $|x|$ function by $\sqrt{x^2+\mu^2}$ as follows:

$$f_2(x_1, x_2; \mu) = \frac{(x_1 + x_2) + \sqrt{(x_1 - x_2)^2 + \mu^2}}{2} \tag{3}$$

Note that as $\mu \to \infty$, $f_1(x_1, x_2; \mu) \to \max(x_1, x_2)$ and as $\mu \to 0$, $f_2(x_1, x_2; \mu) \to \max(x_1, x_2)$. For particular values of x_1 and x_2, we can approximate known activation functions. For example, consider $x_1 = ax$, $x_2 = bx$, with $a \neq b$ in (2), we get:

$$f_1(ax, bx; \mu) = \frac{(a + b)x + (a - b)x\,\text{erf}(\mu(a - b)x)}{2}. \tag{4}$$

This is a simple case from the Maxout family [10] while more complicated cases can be found by considering non-linear choices of x_1 and x_2. We can similarly get smooth approximations to ReLU and Leaky ReLU. For example, consider $x_1 = x$ and $x_2 = 0$, we have smooth approximation of ReLU as follows:

$$f_1(x, 0; \mu) = \frac{x + x\,\text{erf}(\mu x)}{2}. \tag{5}$$

We know that GELU [14] is a smooth approximation of ReLU. Notice that, if we choose $\mu = \frac{1}{\sqrt{2}}$ in equation (5), we can recover GELU activation function which also show that GELU is smooth approximation of ReLU. Also, considering $x_1 = x$ and $x_2 = \alpha x$, we have a smooth approximation of Leaky ReLU or Parametric ReLU depending on whether α is a hyperparameter or a learnable parameter.

$$f_1(x, \alpha x; \mu) = \frac{(1 + \alpha)x + (1 - \alpha)x\,\text{erf}(\mu(1 - \alpha)x)}{2}. \tag{6}$$

Note that, equation (5) and equation (6) approximate ReLU or Leaky ReLU from below. Similarly, we can derive approximating function from equation (3) which will approximate ReLU or Leaky ReLU from above.

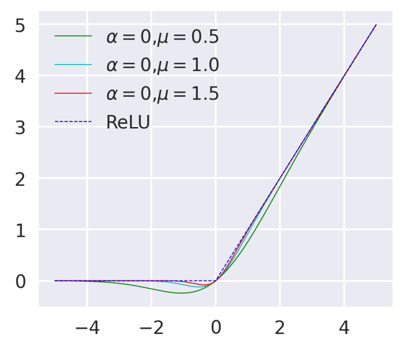

Figure 1. Approximation of ReLU using SMU ($\alpha = 0$) for different values of μ. As $\mu \to \infty$, SMU smoothly approximate ReLU

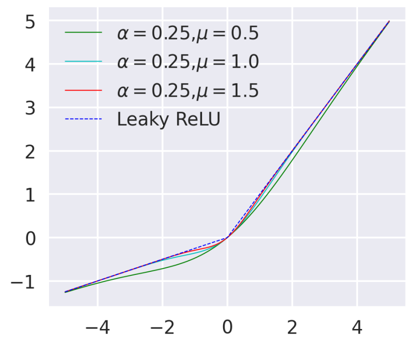

Figure 2. Approximation of Leaky ReLU ($\alpha = 0.25$) using SMU for different values of μ. As $\mu \to \infty$, SMU smoothly approximate Leaky ReLU

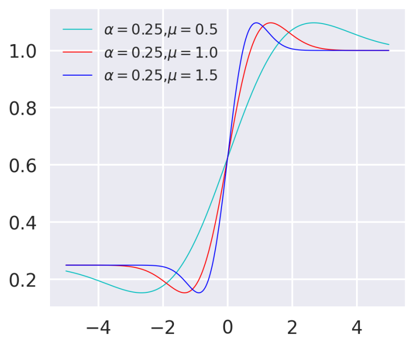

Figure 3. First order derivatives of SMU for $\alpha = 0.25$ and different values of μ.

The corresponding derivatives of equation (6) for input variable x is

$$\frac{d}{dx}f_1(x, \alpha x; \mu) = \frac{1}{2}[(1+\alpha) + (1-\alpha)\,\mathrm{erf}(\mu(1-\alpha)x) + \frac{2}{\sqrt{\pi}}\mu(1-\alpha)^2 x e^{-(\mu(1-\alpha)x)^2}] \quad (7)$$

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

Figures 1, 2, and 3 show the plots for $f_1(x, 0; \mu)$, $f_1(x, 0.25x; \mu)$, and derivative of $f_1(x, 0.25x; \mu)$ for different values of μ. From the figures it is clear that as $\mu \to \infty$, $f_1(x, \alpha x; \mu)$ smoothly approximate ReLU or Leaky ReLU depending on value of α. We call the function in equation (6) as Smooth Maximum Unit (SMU). Similarly, We can derive a function by replacing $x_1 = x$ and $x_2 = \alpha x$ in equation (3) and we call this function SMU-1. For all of our experiments, we will use SMU and SMU-1 as our proposed activation functions.

3.2. Learning activation parameters via back-propagation

Trainable activation function parameters are updated using backpropagation [27] technique (see [12]) according to (8) and for a single layer, the gradient of a hyper-parameter ω is:

$$\frac{\partial L}{\partial \omega} = \sum_x \frac{\partial L}{\partial f(x)} \frac{\partial f(x)}{\partial \omega} \quad (8)$$

where L is the objective function, $\omega \in \{\alpha, \mu\}$ and $f(x) \in \{f_1(x, \alpha x; \mu), f_2(x, \alpha x; \mu)\}$. We implemented forward pass in both Pytorch [39] & Tensorflow-Keras [5] API, and automatic differentiation will update the parameters. Alternatively, CUDA [38] based implementation (see [33]) can be used and the gradients of equations (9) and (10)) for the parameters α and μ of equation (6) can be computed as follows:

$$\frac{\partial f_1}{\partial \alpha} = \frac{x}{2} - \frac{x\,\mathrm{erf}(\mu(1-\alpha)x)}{2} - \frac{(1-\alpha)\mu x^2 e^{-(\mu(1-\alpha)x)^2}}{\sqrt{\pi}} \quad (9)$$

$$\frac{\partial f_1}{\partial \mu} = \frac{1}{\sqrt{\pi}}(1-\alpha)^2 x^2 e^{-(\mu(1-\alpha)x)^2} \quad (10)$$

α and μ can be either hyperparameters or trainable parameters.
Now, note that the class of neural networks with SMU and SMU-1 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 [35]).

Proposition 1. (Theorem 1.1 in Kidger and Lyons, 2020 [20]) :- 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

We report a detailed experimental evaluation in the next subsections on four different deep learning problems like image classification, object detection, semantic segmentation, and machine translation. To compare performance of our proposed activation function, we consider ten popular activation functions as the baseline functions. The following activations are considered to compare performance with SMU and SMU-1: ReLU [36], Leaky ReLU [33], ReLU6

[24], Parametric ReLU (PReLU) [12], ELU [6], Softplus [56], Swish [41], Mish [34], GELU [14], and Pade Activation Unit (PAU) [35]. For all experiments, we consider Swish $(x.\mathrm{Sigmoid}(\beta x))$, PReLU $(\max(\mathrm{x}, \mathrm{ax}))$, and PAU as trainable activation functions. We initialize the trainable parameter β at 1.0 for Swish, a at 0.25 for PReLU. PAU function has ten trainable parameters and all the parameters are initialized as suggested in [35]. All the trainable parameters are updated via the backpropagation [27] algorithm. We report results for baseline activation functions, SMU and SMU-1 activation functions in the following sections. SMU-1 is a computationally cheap activation function due to its simple form, while it boosts the network performance remarkably well in all the experiments compared to the baseline activations. All the experiments are conducted on an NVIDIA Tesla V100 GPU with 32GB RAM.

4.1. Image Classification

We report results for the image classification problem on six popular benchmarking datasets: MNIST, Fashion MNIST, SVHN, CIFAR10, CIFAR100, and Tiny ImageNet. Detailed results are reported in the following subsections. For SMU, we consider $\alpha = 0.25$, a constant hyperparameter and μ as a trainable parameter and initialise at 1.0.

4.1.1 MNIST, Fashion MNIST, and SVHN

In this section, We present our experimental comparison for SMU, SMU-1 and other baseline activations on MNIST ([29]), Fashion MNIST ([51]), and SVHN ([37]) datasets. The MNIST and Fashion MNIST databases contain 60k training and 10k testing 28×28 grey-scale images. Both the datasets have ten different classes. The SVHN database has 32×32 RGB images and a total of 73257 training images and 26032 testing images with ten different classes. Standard data augmentation methods like zoom, rotation, height shift, shearing are applied to these three datasets. We consider a batch size of 128, 0.01 initial learning rate and decay the learning rate with cosine annealing ([31]) learning rate scheduler. We use stochastic gradient descent ([42], [21]) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 100 epochs. We report results with VGG-16 [45] (with batch-normalization [19]) architecture in Table 1 (for more detailed experimental results, see supplemental material) for mean of 15 different runs.

4.1.2 CIFAR

In this section, we report results on the popular image classification datasets CIFAR10 [23] and CIFAR100 [23]. Both the datasets have 50k training and 10k testing images. While CIFAR10 has ten classes and CIFAR100 has 100 classes. In these two datasets for all experiments, we consider a batch size of 128, 0.01 initial learning rate and decay the learning rate with cosine annealing ([31]) learning rate scheduler, stochastic gradient descent ([42], [21]) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 200 epochs. We consider standard data augmentation methods like horizontal flip and rotation. Top-1 accuracy is reported in Table 2 and Table 3 (for more detailed experimental results, please see supplemental material) on CIFAR10 [23] and CIFAR100 [23] datasets respectively for mean of 15 different runs. The results are reported with MobileNet V1 [15], MobileNet V2 [44], ShuffleNet V1 [55] (SF V1), ShuffleNet V2 [32], PreActResNet [13], ResNet [11], GoogleNet [47], Inception V3 [48], DenseNet [17], Squeeze-and-Excitation Networks (SeNet) [16], SqueezeNet [18], ResNext [52], WideResNet [54], Xception [4], VGG [45] (with batch-normalization [19]), AlexNet [25], LeNet [28], and EfficientNet B0 [49]. From Table 2 it is clear that Top-1 classification accuracy improves by 6.19%, 6.22%, 3.39%, 3.51%, 3.09%, 3.40% and 3.08% when we replace ReLU by SMU on the CIFAR100 dataset with ShuffleNet V2 (1.0x), ShuffleNet V2 (2.0x), PreActResNet-50, ResNet-50, ResNext, Xception and SeNet-50 models respectively. The Figures 4 and 5 shows the learning curves on CIFAR100 dataset with ShuffleNet V2 (2.0x) model for the baseline and the proposed activation functions.

4.1.3 Tiny Imagenet

In this section, We report results for classification problem on a more challenging dataset, Tiny Imagenet [26]. Tiny imagenet has RGB images of size 64×64 with total 1,00,000 training images, 10,000 validation images, and 10,000 test images with total 200 classes. Standard data augmentation methods like rotation, horizontal flip is applied. We consider a batch size of 64, 0.1 initial learning rate and reduce the learning rate after every 50 epochs by a factor of 10. We use stochastic gradient descent ([42], [21]) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 200 epochs. Results are reported with WideResNet 28-10 (WRN 28-10) [54], DenseNet-121 [17], ResNet-18, and ResNet-50 [11] models and Top-1 classification accuracy is reported in table 4 for mean of 10 different runs. The proposed functions performs better than the baseline functions and results are stable (mean$\pm$std) and we get very good improvement over the baseline activation functions. Replacing ReLU by SMU, we have 2.56%, 2.23%, 2.31%, and 2.78% boost in Top-1 classification accuracy on DenseNet-121, ResNet-18, ResNet-50, and WideResNet 28-10 models respectively.

4.2. Object Detection

In this section, we report results on object detection problem on Pascal VOC dataset [9] with Single Shot Multi-

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.53 ± 0.07	93.79 ± 0.15	95.97 ± 0.14
Leaky ReLU	99.58 ± 0.08	93.80 ± 0.15	96.02 ± 0.15
PReLU	99.55 ± 0.07	93.90 ± 0.17	96.10 ± 0.16
ReLU6	99.59 ± 0.06	93.93 ± 0.12	96.11 ± 0.15
ELU	99.48 ± 0.05	93.87 ± 0.16	96.05 ± 0.17
Softplus	99.22 ± 0.14	93.58 ± 0.18	95.81 ± 0.21
Swish	99.57 ± 0.05	94.17 ± 0.11	96.20 ± 0.12
Mish	99.63 ± 0.04	94.25 ± 0.13	96.31 ± 0.12
GELU	99.59 ± 0.04	94.22 ± 0.14	96.21 ± 0.14
PAU	99.55 ± 0.07	94.09 ± 0.14	96.20 ± 0.14
SMU	**99.69** ± 0.04	**94.48** ± 0.10	**96.59** ± 0.11
SMU-1	**99.65** ± 0.04	**94.37** ± 0.14	**96.43** ± 0.14

Table 1. Comparison between SMU, SMU-1 activations and other baseline activations on MNIST, Fashion MNIST, and SVHN datasets for image classification problem on VGG16 architecture. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean$\pm$std is reported in the table.

Model	ReLU	SMU	SMU-1
	Top-1 accuracy	Top-1 accuracy	Top-1 accuracy
Shufflenet V2 0.5x	62.07 ± 0.26	**66.67** ± 0.24	**65.60** ± 0.24
Shufflenet V2 1.0x	64.41 ± 0.25	**70.60** ± 0.21	**69.96** ± 0.22
Shufflenet V2 1.5x	67.20 ± 0.26	**72.68** ± 0.19	**72.05** ± 0.20
Shufflenet V2 2.0x	67.52 ± 0.25	**73.74** ± 0.20	**73.45** ± 0.23
PreActResNet 18	73.18 ± 0.22	**76.07** ± 0.20	**75.72** ± 0.22
PreActResNet 34	73.41 ± 0.24	**76.21** ± 0.20	**75.87** ± 0.21
PreActResNet 50	73.89 ± 0.23	**77.28** ± 0.17	**76.85** ± 0.20
ResNet 18	73.23 ± 0.26	**75.22** ± 0.20	**74.91** ± 0.20
ResNet 34	73.33 ± 0.27	**75.77** ± 0.20	**75.59** ± 0.21
ResNet 50	74.12 ± 0.24	**77.63** ± 0.20	**76.89** ± 0.23
SeNet 18	74.77 ± 0.22	**76.17** ± 0.17	**75.44** ± 0.20
SeNet 34	75.12 ± 0.22	**76.79** ± 0.18	**75.79** ± 0.21
SeNet 50	76.09 ± 0.20	**79.17** ± 0.16	**78.45** ± 0.20
ResNext	74.43 ± 0.22	**77.52** ± 0.18	**77.03** ± 0.21
MobileNet V1	71.10 ± 0.26	**73.59** ± 0.22	**73.10** ± 0.22
MobileNet V2	74.17 ± 0.24	**76.31** ± 0.19	**76.03** ± 0.19
Xception	71.22 ± 0.26	**74.62** ± 0.23	**74.11** ± 0.23
EffitientNet B0	76.60 ± 0.27	**79.10** ± 0.22	**78.77** ± 0.23

Table 2. Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR100 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean$\pm$std is reported in the table.

Box Detector(SSD) 300 model [30] and we consider VGG-16 (with batch-normalization) [45] as the backbone network. We use VOC2007 & VOC2012 as train data and VOC2007 as the test dataset. The dataset contains 20 different objects. We consider a batch size of 8, 0.001 initial learning rate and decay the learning rate as reported in [30]. We use SGD ([42], [21]) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained networks up-to 120000 iterations. We do not consider any pre-trained weight. We report the mean average precision (mAP) in Table 5 for the mean of 10 different runs. Replacing ReLU by SMU, we got a 1% improvement in mAP in the test dataset.

Model	ReLU	SMU	SMU-1
	Top-1 accuracy	Top-1 accuracy	Top-1 accuracy
ShuffleNet V2 0.5x	88.40 ± 0.22	**90.63** ± 0.16	**90.39** ± 0.18
ShuffleNet V2 1.0x	90.81 ± 0.24	**92.72** ± 0.18	**92.42** ± 0.20
ShuffleNet V2 1.5x	91.21 ± 0.22	**93.42** ± 0.17	**92.27** ± 0.18
ShuffleNet V2 2.0x	91.70 ± 0.20	**93.61** ± 0.14	**93.40** ± 0.16
PreActResNet 18	93.57 ± 0.20	**94.63** ± 0.15	**94.52** ± 0.17
PreActResNet 34	94.21 ± 0.17	**95.12** ± 0.13	**94.93** ± 0.14
PreActResNet 50	94.30 ± 0.18	**95.37** ± 0.11	**94.94** ± 0.12
ResNet 18	94.10 ± 0.20	**94.78** ± 0.17	**94.51** ± 0.19
ResNet 34	94.22 ± 0.18	**94.91** ± 0.16	**94.77** ± 0.17
ResNet 50	94.26 ± 0.18	**95.38** ± 0.16	**94.92** ± 0.17
SeNet 18	94.29 ± 0.20	**94.75** ± 0.17	**94.56** ± 0.19
SeNet 34	94.42 ± 0.20	**95.27** ± 0.15	**94.89** ± 0.17
SeNet 50	94.55 ± 0.19	**95.92** ± 0.12	**95.22** ± 0.17
ResNext	93.37 ± 0.18	**94.52** ± 0.15	**94.04** ± 0.18
MobileNet V1	92.41 ± 0.14	**93.81** ± 0.11	**93.47** ± 0.11
MobileNet V2	94.22 ± 0.15	**95.50** ± 0.09	**95.27** ± 0.10
Xception	90.51 ± 0.22	**93.25** ± 0.17	**92.59** ± 0.20
EffitientNet B0	95.10 ± 0.15	**96.23** ± 0.10	**96.11** ± 0.12

Table 3. Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR10 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table.

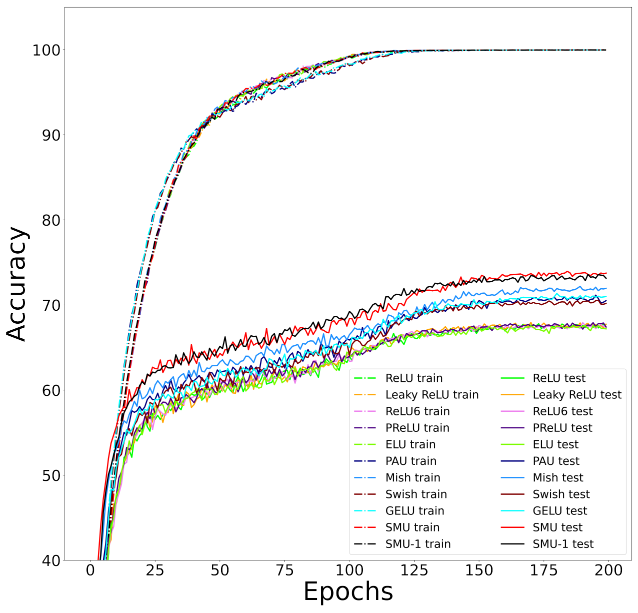

Figure 4. Top-1 train and test accuracy curves for SMU, SMU-1 and other baseline activation functions on CIFAR100 dataset with ShuffleNet V2 (2.0x) model.

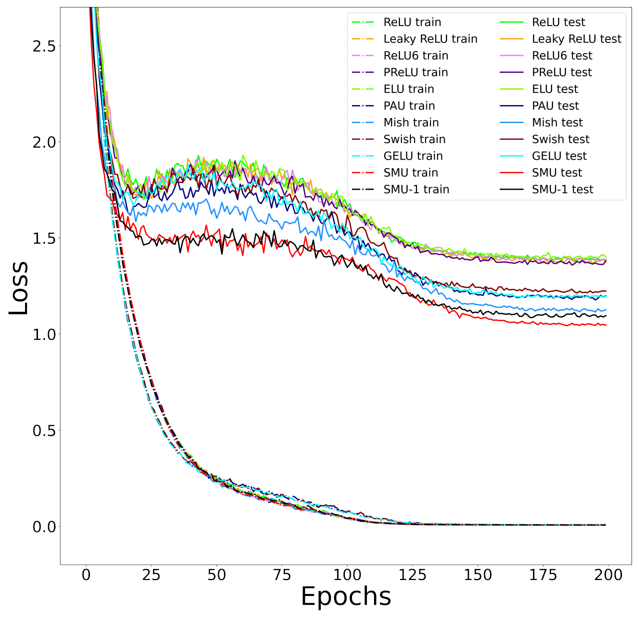

Figure 5. Top-1 train and test loss curves for SMU, SMU-1 and other baseline activation functions on CIFAR100 dataset with ShuffleNet V2 (2.0x) model.

Activation Function	DenseNet-121	ResNet-18	ResNet-50	WideResNet 28-10
ReLU	63.31 $\pm$ 0.47	59.12 $\pm$ 0.44	61.23 $\pm$ 0.46	63.74 $\pm$ 0.40
Leaky ReLU	63.63 $\pm$ 0.48	59.40 $\pm$ 0.44	61.29 $\pm$ 0.44	63.61 $\pm$ 0.42
PReLU	63.71 $\pm$ 0.46	59.59 $\pm$ 0.42	61.35 $\pm$ 0.44	63.78 $\pm$ 0.44
ReLU6	63.54 $\pm$ 0.49	59.49 $\pm$ 0.46	61.41 $\pm$ 0.44	63.72 $\pm$ 0.43
ELU	63.51 $\pm$ 0.46	59.34 $\pm$ 0.44	61.49 $\pm$ 0.43	63.72 $\pm$ 0.43
Softplus	63.01 $\pm$ 0.57	59.01 $\pm$ 0.57	60.93 $\pm$ 0.57	63.01 $\pm$ 0.59
Swish	64.21 $\pm$ 0.40	60.05 $\pm$ 0.40	61.79 $\pm$ 0.41	64.58 $\pm$ 0.41
Mish	64.47 $\pm$ 0.40	60.21 $\pm$ 0.39	62.07 $\pm$ 0.42	64.79 $\pm$ 0.38
GELU	64.34 $\pm$ 0.42	60.21 $\pm$ 0.41	61.66 $\pm$ 0.42	64.39 $\pm$ 0.40
PAU	64.04 $\pm$ 0.43	60.37 $\pm$ 0.39	61.72 $\pm$ 0.41	64.42 $\pm$ 0.40
SMU	**65.87** $\pm$ 0.37	**61.35** $\pm$ 0.35	**63.54** $\pm$ 0.40	**66.52** $\pm$ 0.35
SMU-1	**65.09** $\pm$ 0.38	**60.93** $\pm$ 0.38	**62.79** $\pm$ 0.40	**65.25** $\pm$ 0.37

Table 4. Comparison between SMU, SMU-1 activations and other baseline activations on Tiny ImageNet dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 10 different runs. mean$\pm$std is reported in the table.

Activation Function	mAP
ReLU	77.2 $\pm$ 0.14
Leaky ReLU	77.2 $\pm$ 0.13
PReLU	77.2 $\pm$ 0.16
ReLU6	77.1 $\pm$ 0.15
ELU	75.1 $\pm$ 0.18
Softplus	74.2 $\pm$ 0.25
Swish	77.5 $\pm$ 0.11
Mish	77.6 $\pm$ 0.11
GELU	77.5 $\pm$ 0.12
PAU	77.4 $\pm$ 0.14
SMU	**78.2** $\pm$ 0.09
SMU-1	**77.8** $\pm$ 0.11

Table 5. Comparison between SMU, SMU-1 activations and other baseline activations on Pascal VOC dataset for object detection problem. We report mAP for the mean of 10 different runs. mean$\pm$std is reported in the table.

Activation Function	Pixel Accuracy	mIOU
ReLU	79.49 $\pm$ 0.46	69.31 $\pm$ 0.28
Leaky ReLU	79.41 $\pm$ 0.41	69.64 $\pm$ 0.42
PReLU	78.95 $\pm$ 0.42	68.88 $\pm$ 0.41
ReLU6	79.58 $\pm$ 0.41	69.70 $\pm$ 0.42
ELU	79.48 $\pm$ 0.50	68.19 $\pm$ 0.40
Softplus	78.45 $\pm$ 0.52	68.08 $\pm$ 0.49
Swish	80.22 $\pm$ 0.46	69.81 $\pm$ 0.30
Mish	80.59 $\pm$ 0.44	70.12 $\pm$ 0.30
GELU	80.14 $\pm$ 0.37	69.59 $\pm$ 0.40
PAU	79.89 $\pm$ 0.39	69.31 $\pm$ 0.44
SMU	**81.79** $\pm$ 0.36	**71.11** $\pm$ 0.30
SMU-1	**80.75** $\pm$ 0.41	**70.55** $\pm$ 0.30

Table 6. Comparison between SMU, SMU-1 activations and other baseline activations on CityScapes dataset for semantic segmentation problem. We report pixel accuracy and mIOU for the mean of 10 different runs. mean$\pm$std is reported in the table.

4.3. Semantic Segmentation

In this section, we report experimental results on semantic segmentation problems on the popular CityScapes dataset [7]. CityScapes [7] is a popular dataset consisting of diverse urban street scenes across 50 different cities at varying times of the year, as well as ground truths for semantic segmentation, instance-level segmentation. Label annotations for segmentation tasks span across 30+ classes. We consider U-net model [43] as the segmentation framework. The model is trained with adam optimizer [22], $5e^{-3}$ learning rate, a batch size 32 up to 250 epochs. We report the mean of 10 different runs for Pixel Accuracy and the mean Intersection-Over-Union (mIOU) on test data on table 6.

4.4. Machine Translation

In this section, we report the result on the machine translation problem. This problem deals with the translation of text or speech data from one language to another language without the help of any human being. The WMT 2014 English$\rightarrow$German dataset is used for our experiment. The database contains 4.5 million training sentences. We use an attention-based [50] 8-head transformer network with Adam optimizer [22], 0.1 dropout rate [46], and train up to 100000 steps. Other hyperparameters are kept similar as mentioned in the original paper [50]. We evaluate the network performance on the newstest2014 dataset using the BLEU score metric. We report the mean of 10 dif-

Baselines	ReLU	Leaky ReLU	ELU	Softplus	PReLU	ReLU6	Swish	Mish	GELU	PAU
SMU $>$ Baseline	80	80	80	80	80	80	77	76	77	78
SMU $=$ Baseline	0	0	0	0	0	0	0	0	0	0
SMU $<$ Baseline	0	0	0	0	0	0	3	4	3	2

Table 7. Baseline table for SMU. These numbers represent the total number of models in which SMU underperform, equal or outperform compared to the baseline activation functions

ferent runs on Table 8 on the test dataset(newstest2014). The table shows that the results are stable on different runs (mean$\pm$std), and we got around 0.6% boost in BLEU score for SMU compared to ReLU.

Activation Function	BLEU Score
ReLU	26.2 ± 0.14
Leaky ReLU	26.3 ± 0.15
PReLU	26.2 ± 0.18
ReLU6	26.1 ± 0.14
ELU	25.1 ± 0.14
Softplus	23.6 ± 0.18
Swish	26.4 ± 0.11
Mish	26.3 ± 0.12
GELU	26.4 ± 0.15
PAU	26.3 ± 0.15
SMU	$\mathbf{26.8} \pm 0.11$
SMU-1	$\mathbf{26.6} \pm 0.10$

Table 8. Comparison between SMU, SMU-1 activations and other baseline activations on WMT2014 dataset for machine translation problem. We report BLEU score for the mean of 10 different runs. mean$\pm$std is reported in the table.

5. Baseline Table

SMU and SMU-1 are novel activation functions constructed using the smoothing of maximum function. For a detailed comparison, we report a summary of all the experiments in Table 7 given in earlier sections and the supplementary material. It is pretty clear from Table 7 that the proposed functions outperform baseline functions almost in all experiments.

6. Computational Time Comparison

In this section, we report the computational Time Comparison for SMU, SMU-1, and baseline activation functions. We report results in Table 9 for the mean of 100 runs on a 32×32 RGB image in ResNet-18 [11] model for both forward and backward pass. The experiments are conducted on an NVIDIA Tesla V100 GPU with 32GB RAM. It is noticeable from the experiment section and Table 9 that there is a small trade-off between the computational time and model performances compared to ReLU or its variants. The proposed activations have significantly boosted the model performance though it has slightly higher computational time (due to non-linearity and the trainable parameter μ) than ReLU or its variants. In contrast, the computational time is similar to popular non-linear activations like Swish, Mish & GELU and much better than PAU, while model performance at the same time is comparatively much better than these four popular non-linear activations in almost all cases.

Activation Function	Forward Pass	Backward Pass
ReLU	$6.43 \pm 0.31\ \mu s$	$6.28 \pm 0.74\ \mu s$
Leaky ReLU	$6.49 \pm 0.41\ \mu s$	$6.41 \pm 0.95\ \mu s$
PReLU	$8.20 \pm 1.57 \mu s$	$9.26 \pm 1.86\ \mu s$
ReLU6	$6.45 \pm 0.45\ \mu s$	$6.41 \pm 0.91\ \mu s$
ELU	$6.51 \pm 0.50\ \mu s$	$6.42 \pm 0.88\ \mu s$
Softplus	$6.49 \pm 0.49\ \mu s$	$6.40 \pm 0.55\ \mu s$
Mish	$10.02 \pm 1.79\ \mu s$	$11.97 \pm 1.75\ \mu s$
GELU	$10.75 \pm 1.49\ \mu s$	$12.49 \pm 1.77\ \mu s$
Swish	$10.47 \pm 1.10\ \mu s$	$12.61 \pm 1.22\ \mu s$
PAU	$18.45 \pm 3.40\ \mu s$	$25.99 \pm 5.06\ \mu s$
SMU	$10.74 \pm 1.29\ \mu s$	$12.95 \pm 1.54\ \mu s$
SMU-1	$9.68 \pm 1.81\ \mu s$	$11.98 \pm 1.49\ \mu s$

Table 9. Runtime comparison for the forward and backward passes for SMU and SMU-1 and other baseline activation functions for a $32\times$ 32 RGB image in ResNet-18 model.

7. Conclusion

This work uses the maximum smoothing technique to approximate Leaky ReLU, a well-established activation function (not differentiable at 0) by two smooth functions. We call these two functions SMU and SMU-1, and we use them as potential candidates for activation functions. Our experimental evaluation shows that the proposed functions beat the traditional activation functions in well-known deep learning problems and have the potential to replace them.

References

[1] Koushik Biswas, Shilpak Banerjee, and Ashish Kumar Pandey. Orthogonal-padé activation functions: Trainable activation functions for smooth and faster convergence in deep networks, 2021. 1

[2] Koushik Biswas, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. Erfact and pserf: Non-monotonic smooth trainable activation functions, 2021. 1

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. 1

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017. 4

[5] François Chollet et al. Keras. `https://keras.io`, 2015. 3

[6] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2016. 1, 4

[7] 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. 7

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. 1

[9] 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. 4

[10] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks, 2013. 2

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 4, 8

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015. 1, 3, 4

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks, 2016. 4

[14] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2020. 1, 2, 4

[15] 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. 4

[16] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019. 4

[17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2016. 4

[18] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size, 2016. 4

[19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 4

[20] Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks, 2020. 3

[21] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952. 4, 5

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 7

[23] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 4

[24] Alex Krizhevsky. Convolutional deep belief networks on cifar-10, 2010. 4

[25] 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, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc. 4

[26] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. 2015. 4

[27] 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. 3, 4

[28] 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. 4

[29] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010. 4

[30] 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*, page 21–37, 2016. 5

[31] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. 4

[32] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design, 2018. 4

[33] 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. 1, 3

[34] Diganta Misra. Mish: A self regularized non-monotonic activation function, 2020. 1, 4

[35] Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Padé activation units: End-to-end learning of flexible activation functions in deep networks, 2020. 1, 3, 4

[36] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814. Omnipress, 2010. 1, 3

[37] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 4

[38] J. Nickolls, I. Buck, M. Garland, and K. Skadron. Scalable parallel programming. In *2008 IEEE Hot Chips 20 Symposium (HCS)*, pages 40–53, 2008. 3

[39] 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. 3

[40] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 1

[41] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017. 1, 4

[42] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951. 4, 5

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 7

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019. 4

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 4, 5

[46] 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, Jan. 2014. 7

[47] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014. 4

[48] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015. 4

[49] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. 4

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 7

[51] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 4

[52] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2017. 4

[53] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network, 2015. 1

[54] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016. 4

[55] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017. 4

[56] 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)*, pages 1–4, 2015. 1, 4