# An Improved Activation Function Based on Power Linear Units (Postprint)

**Authors:** Luo Xunhao, Li Peihua

**Date:** 2018-08-13T00:00:00+00:00

## Abstract

Nonlinear activation functions play a critical role in convolutional neural networks. Addressing the issue that the Rectified Linear Unit (ReLU) completely discards negative activation values containing useful information in the network, we propose a novel parametric activation function called Power Linear Unit (PoLU) based on research on Parametric Rectified Linear Unit (PReLU) and Exponential Linear Unit (ELU). PoLU applies a signed power nonlinear transformation to the negative activation portion of the input, where the parameters of the power function can be adaptively learned during the training process of CNNs, while simultaneously keeping the positive activation portion unchanged like ReLU. PoLU can be efficiently implemented and flexibly applied to different convolutional neural network architectures. Experimental results on the widely used CIFAR-10/100 datasets demonstrate that PoLU outperforms ReLU and its corresponding activation functions.

## Full Text

## Preamble

### Improved Activation Function Based on Power Linear Unit

*Luo Xunhao, Li Peihua*
*Faculty of Electronic Information & Electrical Engineering, Dalian University of Technology, Dalian, Liaoning 116024, China*

**Abstract:** Non-linear activation functions play a critical role in convolutional neural networks (CNNs). To address the issue that the rectified linear unit (ReLU) completely discards negative activations which often contain useful information, this paper proposes a novel parametric activation function called the Power Linear Unit (PoLU), building upon research into the Parametric Rectified Linear Unit (PReLU) and Exponential Linear Unit (ELU). PoLU applies a signed power non-linear transformation to negative activations, where the power

function parameters can be learned adaptively during CNN training while keeping positive activations unchanged, similar to ReLU. PoLU can be implemented efficiently and flexibly integrated into various CNN architectures. Experimental results on the widely-used CIFAR-10/100 benchmarks demonstrate that PoLU outperforms ReLU and its corresponding activation function counterparts.

**Keywords:** power linear unit; parametric activation function; convolutional neural networks

## 1. Activation Functions

Deep convolutional neural networks (CNNs) have attracted significant attention and achieved remarkable performance in many computer vision tasks [8,17,18]. Activation functions are fundamental components in CNNs, and recent research indicates that using the rectified linear unit (ReLU) as the activation function is key to the success of deep CNNs [8,15,10,9]. ReLU was first proposed for restricted Boltzmann machines [7] and later successfully applied to neural networks [1]. The definition of ReLU is $\max(0, x)$. Compared to traditional Sigmoid functions, ReLU offers two major advantages [2]. First, ReLU can mitigate the vanishing gradient problem [6] while accelerating convergence and preventing the network from converging to a local optimum. Second, ReLU tends to produce sparse representations, which typically lead to better classifier performance [8].

Despite these benefits, ReLU still has several drawbacks. One major disadvantage is that ReLU ignores negative activations, which often contain much useful information for representing targets, particularly in the shallow layers of deep CNNs [3]. To overcome this limitation, many improved variants of ReLU have been proposed. Table 1 summarizes existing methods. Leaky ReLU (LReLU) [2] modifies ReLU by modeling negative activations as a linear function, defined as $\max(0, x) + \min(0, ax)$ with $a = 0.01$. LReLU enables negative activations to propagate throughout the deep CNN by multiplying them by a small scalar such as 0.01. Since LReLU has non-zero derivatives for negative activations, parameters with negative activation values can also be updated during end-to-end learning. Experimental results show that LReLU achieves better classification accuracy than ReLU. However, LReLU uses the same parameter throughout the entire network, which is an unreasonable setting because negative activations serve different purposes in different layers of deep CNNs.

To address this issue, He et al. [3] proposed the Parametric Rectified Linear Unit (PReLU). This activation function introduces a parameterized linear transformation for negative activations, and its parameters can be updated simultaneously with the original deep network parameters through backpropagation. PReLU is defined as $\max(0, x) + \min(0, ax)$, where $a$ is a learnable parameter that controls the nonlinearity of the negative activation part. He et al. demonstrated that automatically learning parameter $a$ outperforms manual tuning, as in LReLU.

Unlike LReLU and PReLU, another recently proposed activation function is the Exponential Linear Unit (ELU) [4], which exhibits nonlinear transformation for negative values: $\max(0, x) + \min(0, \beta(\exp(x) - 1))$ with $\beta > 0$. The parameter $\beta$ in ELU is typically set manually, often to 1. The nonlinear transformation defined in the negative activation region of ELU can reduce bias shift, bringing the standard gradient closer to the natural gradient and thus accelerating training. Experimental results show that ELU outperforms other activation functions on various vision tasks. However, similar to LReLU, ELU applies the same nonlinear transformation to negative activations across all layers of deep CNNs, which is inappropriate for practical scenarios.

Based on the above discussion and inspired by PReLU and ELU, this paper proposes a novel parametric activation function called the Power Linear Unit (PoLU). As shown in Table 1 and Figure 1 [Figure 1: see original paper], unlike existing activation functions, the proposed PoLU can present different forms in different layers of deep CNNs by introducing a learnable parameter. Additionally, PoLU can be efficiently implemented and flexibly applied to existing deep CNNs. Our experiments are conducted on the widely-used CIFAR-10 and CIFAR-100 datasets, and the results demonstrate that PoLU outperforms other corresponding activation functions on the deep convolutional neural network architecture Network in Network [10].

---

## 2. Power Linear Unit

This section details the proposed Power Linear Unit (PoLU). Two types of PoLU are introduced: channel-shared and channel-wise (channel-exclusive).

### 2.1 Forward Propagation

The channel-wise PoLU is defined as follows:

$$f_i(x_i) = \begin{cases} x_i & \text{if } x_i > 0 \\ -\text{sign}(x_i) \cdot |x_i|^{a_i} & \text{if } x_i \leq 0 \end{cases}$$

where $x_i$ represents the input to the nonlinear activation function $f(\cdot)$ for the $i$-th channel, and $a_i$ is a learnable parameter that controls the nonlinearity of the negative activation part. Here we restrict the learned parameter $a_i > 0$. When $a_i = 0.5$, PoLU behaves as a square root function. The smaller the $a_i$, the smaller the resulting negative activation values, and the negative part stays closer to the x-axis; conversely, larger values make the negative part farther from the x-axis. PoLU controls the nonlinear behavior of the negative activation part through this characteristic. Figure 1 shows a comparison between PoLU and other activation functions when $a_i = 0.5$. By utilizing the learnable parameter $a_i$, deep CNNs can control their nonlinear behavior during the training phase.

According to the PoLU definition in Equation (1), smaller $a_i$ values introduce more nonlinearity.

### 2.2 Backpropagation

PoLU can be trained using the backpropagation algorithm to update its parameters. According to the chain rule, the derivative of the loss function $E$ with respect to parameter $a_i$ is:

$$\frac{\partial E}{\partial a_i} = \sum_{x_i} \frac{\partial E}{\partial f_i(x_i)} \cdot \frac{\partial f_i(x_i)}{\partial a_i}$$

where $\frac{\partial E}{\partial f_i(x_i)}$ is the gradient propagated from deeper layers of the network, and $\frac{\partial f_i(x_i)}{\partial a_i}$ is the derivative of the activation function $f(\cdot)$ with respect to $a_i$:

$$\frac{\partial f_i(x_i)}{\partial a_i} = \begin{cases} 0 & \text{if } x_i > 0 \\ -\ln|x_i| \cdot |x_i|^{a_i} & \text{if } x_i \leq 0 \end{cases}$$

The parameter update rule is as follows:

$$\Delta a_i \leftarrow \mu \cdot \Delta a_i + \omega \cdot \left( \frac{\partial E}{\partial a_i} + \alpha \cdot a_i \right)$$

$$a_i \leftarrow a_i + \Delta a_i$$

where $\mu$ is the momentum coefficient, $\omega$ is the learning rate, and $\alpha$ is the weight decay coefficient.

Additionally, the derivative of the loss function $E$ with respect to the input $x_i$ is:

$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial f_i(x_i)} \cdot \frac{\partial f_i(x_i)}{\partial x_i}$$

In Equation (5), $\frac{\partial f_i(x_i)}{\partial x_i}$ is the derivative of the activation function $f(\cdot)$ with respect to $x_i$:

$$\frac{\partial f_i(x_i)}{\partial x_i} = \begin{cases} 1 & \text{if } x_i > 0 \\ -a_i \cdot |x_i|^{a_i - 1} & \text{if } x_i \leq 0 \end{cases}$$

### 2.3 Channel-Shared Variant

In channel-wise PoLU, all channel parameters $a_i$ in a layer are different, and the number of additional parameters equals the number of channels in the corresponding network layer. In contrast, the parameter $a$ in channel-shared PoLU is shared across all channels of a network layer, with each PoLU unit introducing only one additional parameter. The backpropagation for channel-shared PoLU is as follows:

$$\frac{\partial E}{\partial a} = \sum_i \sum_{x_i} \frac{\partial E}{\partial f_i(x_i)} \cdot \frac{\partial f_i(x_i)}{\partial a}$$

where $\sum_i$ represents the sum of gradients across all channels in a deep CNN layer. This channel-shared PoLU introduces only one additional parameter per layer, resulting in negligible memory consumption and computational cost. Compared to channel-wise PoLU, channel-shared PoLU introduces fewer parameters and is more computationally efficient. However, as experiments demonstrate, channel-wise PoLU achieves better performance.

---

## 3. Experimental Results and Analysis

This chapter details the experimental setup. We conduct experiments on the typical deep CNN architecture Network in Network (NIN) and compare with four different activation functions: ReLU [1], LReLU [2], PReLU [3], and ELU [4]. Comparative experiments are performed on the CIFAR-10 and CIFAR-100 databases.

### 3.1 Experimental Databases

The CIFAR-10 (C10) database consists of 10 classes of color natural scene images, containing 50,000 training images and 10,000 test images, each a 32×32 RGB image. We use the image preprocessing method shown in [5], which includes color normalization and ZCA whitening. Data augmentation during training follows [10,13,11], where 4 pixels are padded on each side of the original image to create 40×40 padded images, from which 32×32 images are randomly cropped with a 0.5 probability of horizontal flipping. During testing, only color-normalized images are used. Following [11], this augmented dataset is called C10+. The CIFAR-100 (C100) database is identical to C10 except it contains 100 classes of images. The C100+ augmentation method is the same as C10+.

### 3.2 Experimental Results on NIN

The NIN network structure used in this paper follows [10], consisting of three cascaded mlpconv layers, each followed by a spatial pooling layer for downsampling. Each mlpconv layer contains a convolutional layer and two cascaded

cross-channel parametric pooling (cccp) layers, where cccp layers are equivalent to convolutional layers with 1×1 kernels. The last mlpconv layer contains only one convolutional layer and one cccp layer. All layers except the final mlpconv layer use Dropout [14] technology. The last layer of NIN is a global average pooling layer, a k-channel fully connected layer, and a softmax layer.

The training process for networks with PoLU is similar to the typical AlexNet [8] training procedure. Networks are trained using stochastic gradient descent with a batch size of 128, momentum coefficient of 0.9, and weight decay coefficient of 0.0005. The initial learning rate is 0.04, which is reduced by a factor of 10 after 80 epochs when the validation error rate stabilizes.

**3.2.1 Manual Parameter Setting Analysis** The PoLU activation function has a key parameter $a$ that controls the nonlinear behavior of the negative activation part. This experiment verifies the impact of parameter $a$ on PoLU performance using channel-shared PoLU, where each PoLU layer shares the same parameter to exclude interference from other factors. The baseline test error on C10 is 10.41% [10]. In comparative experiments, we manually set the parameter $a$ for each PoLU layer in NIN to the same value across five groups of experiments, with the evaluation metric being Top-1 test error (%). Table 2 shows the comparative experimental results, indicating that both excessively large and small parameter values cause performance degradation. In these experiments, parameter $a$ was set to the same value for all PoLU layers, meaning all PoLU layers exhibit identical nonlinear behavior—an unreasonable setting since each convolutional layer in a CNN has different nonlinearities and should have different PoLU layer parameters. Manually setting each PoLU layer' s parameter $a$ would be labor-intensive and require extensive experience. This paper introduces $a$ as a learnable parameter that updates adaptively during CNN training, allowing each PoLU layer corresponding to a convolutional layer to learn its most suitable parameter.

**3.2.2 Comparison Between Channel-Shared and Channel-Wise Nonlinearity** First, we train an NIN with ReLU activation on C10 as a baseline, achieving a top-1 error rate of 10.10%. Next, all ReLU activations in NIN are replaced with PoLU activations, and the network is trained from scratch using the same settings as the ReLU-based NIN. The PoLU layers in NIN are channel-wise with parameters $a_i$ initialized to 0.5. Figures 2 [Figure 2: see original paper] and 3 [Figure 3: see original paper] show the training and test errors for these two networks. To more clearly display the convergence trend of test errors, Figure 4 [Figure 4: see original paper] shows the test error curves after the 60th epoch. The results show that PoLU activation functions outperform ReLU by 1.31%. Table 3 compares channel-wise PoLU with channel-shared PoLU, demonstrating that channel-wise PoLU outperforms channel-shared PoLU and also surpasses PReLU. All remaining experiments in this paper use channel-wise PoLU and channel-wise PReLU.

**3.2.3 Comparison with Other Nonlinear Activation Functions** We compare PoLU with four other activation functions: ReLU, LReLU, ELU, and PReLU. Comparative experiments are conducted on four databases: C10, C10+, C100, and C100+. For each database, when training a new network model from scratch, only the activation function is changed while other settings remain constant. The reproduced top-1 test errors for NIN with ReLU activation are 10.10% and 33.02% on C10 and C100, respectively—both better than the results reported in [10]. The reproduced ReLU experiment on C10+ achieves 8.83%, comparable to the baseline in [10].

Table 4 summarizes the comparative experimental results. PoLU demonstrates the best performance among all activation functions. On C10, PoLU improves ReLU performance from 10.10% to 8.79%; on C10+, from 8.83% to 7.84%; on C100, from 33.02% to 31.73%; and on C100+, from 32.51% to 30.64%. In most cases, LReLU, ELU, and PReLU perform better than ReLU, with PReLU achieving the best performance among the four. Compared to PReLU, our PoLU achieves improvements of 0.52%, 0.13%, 0.91%, and 0.77% on C10, C10+, C100, and C100+, respectively.

### 3.3 Analysis of Learned Parameters

As shown in [16], deeper layers in CNNs capture more semantic information (e.g., fully connected layers and the final convolutional layer), while shallow layers are more similar to Gabor filters and sensitive to edges and textures (e.g., the first convolutional layer). Table 5 shows the parameters learned by each PoLU layer in NIN. The results demonstrate that deeper mlpconv layers learn smaller parameter values, indicating that activation functions become more nonlinear as layer depth increases. Compared to deep layers, shallow layers contain more information in their negative activation parts and therefore learn larger parameter values.

---

## 4. Conclusion

This paper proposes a novel parametric activation function for deep CNNs called the Power Linear Unit (PoLU). Unlike existing activation functions, our PoLU can exhibit different nonlinear transformations for negative activations across different layers. Experimental results on CIFAR-10 and CIFAR-100 demonstrate that PoLU improves performance on the NIN network model. Additionally, we show that negative activation parts serve different purposes in different layers, and the design and analysis of PoLU contribute to a better understanding of deep CNNs.

## References

[1] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks [C]// Proc of the 14th International Conference on Artificial Intelligence and Statistics. 2011: 315-323.

[2] Maas A L, Hannun A Y, Ng A Y. Rectifier nonlinearities improve neural network acoustic models [C]// Proc of International Conference on Machine Learning. 2013: 3.

[3] He Kaiming, Zhang Xiangyu, Ren Shaoqing, et al. Delving deep into rectifiers: surpassing human-level performance on imagenet classification [C]// Proc of IEEE International Conference on Computer Vision. 2015: 1026-1034.

[4] Clevert D A, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (elus) [C]// Proc of International Conference on Learning Representations. 2016.

[5] Goodfellow I J, Warde-Farley D, Mirza M, et al. Maxout networks [EB/OL]. (2013-09-20). https://arxiv.org/abs/1302.4389.

[6] Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions [J]. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 1998, 6(2): 107-116.

[7] Nair V, Hinton G E. Rectified linear units improve restricted Boltzmann machines [C]// Proc of the 27th International Conference on Machine Learning. 2010: 807-814.

[8] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks [C]// Advances in Neural Information Processing Systems. 2012: 1097-1105.

[9] He Kaiming, Zhang Xiangyu, Ren Shaoqing, et al. Deep residual learning for image recognition [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2016: 770-778.

[10] Lin Min, Chen Qiang, Yan Shuicheng. Network in network [EB/OL]. (2014-03-04). https://arxiv.org/abs/1312.4400.

[11] Huang Gao, Liu Zhuang, Weinberger K Q, et al. Densely connected convolutional networks [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2017: 3.

[12] Krizhevsky A, Hinton G. Learning multiple layers of features from tiny images [D]. Toronto: University of Toronto, 2009.

[13] Huang Gao, Sun Yu, Liu Zhuang, et al. Deep networks with stochastic depth [C]// Proc of European Conference on Computer Vision. Cham: Springer, 2016: 646-661.

[14] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving neural networks by preventing co-adaptation of feature detectors [EB/OL]. (2012-09-03). https://arxiv.org/abs/1207.0580.

[15] Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift [EB/OL]. (2015-03-20). https://arxiv.org/abs/1502.03167.

[16] Zeiler M D, Fergus R. Visualizing and understanding convolutional networks [C]// Proc of European Conference on Computer Vision. Cham: Springer, 2014: 818-833.

[17] Vinyals O, Toshev A, Bengio S, et al. Show and tell: a neural image caption generator [C]// Proc of Computer Vision and Pattern Recognition. 2015: 3156-3164.

[18] Ren Shaoqing, He Kaiming, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks [C]// Advances in Neural Information Processing Systems. 2015: 91-99.

[19] Jaderberg M, Simonyan K, Zisserman A. Spatial transformer networks [C]// Advances in Neural Information Processing Systems. 2015: 2017-2025.

*Note: Figure translations are in progress. See original paper for figures.*