FReLU: Flexible Rectified Linear Units for Improving Convolutional Neural Networks

Suo Qiu, Xiangmin Xu ∗ and Bolun Cai
School of Electronic and Information Engineering, South China University of Technology
Wushan RD., Tianhe District, Guangzhou, P.R.China
Email: q.suo@foxmail.com, xmxu@scut.edu.cn, caibolun@gmail.com

Abstract—Rectified linear unit (ReLU) is a widely used activation function for deep convolutional neural networks. However, because of the zero-hard rectification, ReLU networks lose the benefits from negative values. In this paper, we propose a novel activation function called flexible rectified linear unit (FReLU) to further explore the effects of negative values. By redesigning the rectified point of ReLU as a learnable parameter, FReLU expands the states of the activation output. When a network is successfully trained, FReLU tends to converge to a negative value, which improves the expressiveness and thus the performance. Furthermore, FReLU is designed to be simple and effective without exponential functions to maintain low-cost computation. For being able to easily used in various network architectures, FReLU does not rely on strict assumptions by self-adaption. We evaluate FReLU on three standard image classification datasets, including CIFAR-10, CIFAR-100, and ImageNet. Experimental results show that FReLU achieves fast convergence and competitive performance on both plain and residual networks.

I. INTRODUCTION

The activation function is an important component in neural networks. It provides the non-linear property for deep neural networks and controls the information propagation through adjacent layers. Therefore, the design of the activation function usually plays an important role to achieve good learning behavior and performance. Different activation functions have different characteristics and are used for different tasks. For example, long short-term memory (LSTM) models [1] use sigmoid or hyperbolic tangent functions, while rectified linear unit (ReLU) [2], [3], [4] is widely used in convolutional neural networks (CNNs). In this paper, we mainly focus on extending the ReLU function to improve CNNs.

ReLU [5] is a classical activation function, of which the effectiveness has been verified in previous works [6], [2], [3], [4]. The success of ReLU owes to identically propagating all the positive inputs, which alleviates gradient vanishing and allows the supervised training of much deeper neural networks. Moreover, ReLU is computational efficient by just outputting zero for negative inputs. Thus, it is widely used in neural networks. Although ReLU is fantastic, researchers found that it is not the end of the story about the activation function – the challenges of the activation function arise from two main aspects: negative missing and zero-like property.

∗Xiangmin Xu is the corresponding author.

Negative missing. ReLU simply restrains the negative value to hard-zero, which provides sparsity but results in negative missing. The variants of ReLU, including leaky ReLU (LReLU) [7], parametric ReLU (PReLU) [8] and randomized ReLU (RReLU) [9] enable non-zero slope to the negative part. These methods suggest that the negative part is helpful for neural networks. However, their unstable rectification will destroy sparsity.

Zero-like property. In [10], authors explained that pushing activation means closer to zero (zero-like) can speed up learning. ReLU is apparently not zero-like. LReLU, PReLU, and RReLU cannot ensure a noise-robust negative deactivation state. To this end, exponential linear unit (ELU) [10] was proposed to keep negative values and saturate the negative part to realize the zero-like property. Recent variants [11], [12], [13], [14], [15] of ELU and the penalized tanh function [16] also demonstrate similar performance improvements. However, the incompatibility between ELU and the batch normalization (BN) [17] method has not been well treated.

In this paper, we propose a novel activation function called flexible rectified linear unit (FReLU), which can adaptively adjust the ReLU output by a learnable rectified point to capture the negative information and provide the zero-like property. We evaluate FReLU on image classification tasks and find that the flexible bias rectification can improve the capacity of neural networks. In summary, the proposed activation function FReLU brings the following benefits:

- fast convergence and competitive performance;
- low computation cost without exponential operation;
- compatibility with batch normalization;
- weak assumptions and self-adaptation.

Fig. 1. Illustration of (a) ReLU and (b) FReLU function.
II. THE PROPOSED METHOD

A. Flexible Rectified Linear Unit

As illustrated in Fig. 1(a), let variable $x$ represent the input. The rectified linear unit (ReLU) [2] is defined as:

$$relu(x) = \begin{cases} 
  x & \text{if } x > 0 \\
  0 & \text{if } x \leq 0 
\end{cases}.$$  

(1)

By redesigning the rectified point of ReLU as a learnable parameter, we propose flexible rectified linear unit (FReLU) to improve flexibility on the horizontal and vertical axes, which is expressed as:

$$frelu(x) = relu(x + a) + b,$$  

(2)

where $a$ and $b$ are two learnable variables. Since the activation function generally follows convolutional/linear layer, the arbitrary input value for FReLU can be learned by the preceding convolutional/linear layer. Thus, the $x + a$ can be equivalent to $x$ and the Equ. (2) equals to

$$frelu(x) = relu(x) + b,$$  

(3)

which is illustrated in Fig. 1(b).

Therefore, the forward pass function of FReLU is rewrite as:

$$frelu(x) = \begin{cases} 
  x + b_l & \text{if } x > 0 \\
  b_l & \text{if } x \leq 0 
\end{cases}.$$  

(4)

where $b_l$ is the $l$-th layer-wise learnable parameter, which controls the output range of FReLU. Note that FReLU naturally generates ReLU when $b_l = 0$.

The backward pass function of FReLU is given by:

$$\frac{\partial frelu(x)}{\partial x} = \begin{cases} 
  1 & \text{if } x > 0 \\
  0 & \text{if } x \leq 0 
\end{cases}.$$  

(5)

B. Parameter Initialization with FReLU

As mentioned in [8], it is necessary to adopt appropriate initialization method for a novel activation function to prevent the vanishing problem of gradients. In this subsection, we provide a brief analysis on the initialization for FReLU. More discussions about the initialization of neural networks can refer to [18], [8].

1) Back propagation: For the back propagation case, the gradient of a convolution layer is computed by: $\frac{\partial \text{Cost}}{\partial x_l} = W_l \frac{\partial \text{Cost}}{\partial \hat{x}_l}$, where $x_l = W_l \hat{x}_l$. $W_l$ is a $c$-by-$n$ matrix which is reshaped from $W_l$. Here, $c$ is the number of channels for the input and $n = k^2 d$ ($k$ is the kernel size, and $d$ is the number of channels for the output). We assume $\hat{n}_l$ $w_l$s and $w_l$ and $\frac{\partial \text{Cost}}{\partial x_l}$ are independent of each other. When $w_l$ is initialized by a symmetric distribution around zero, $\text{Var} \left[ \frac{\partial \text{Cost}}{\partial x_l} \right] = \hat{n}_l \text{Var}[w_l] E \left[ \left( \frac{\partial \text{Cost}}{\partial x_l} \right)_l \right]^2$. And for FReLU, we have: $\frac{\partial \text{Cost}}{\partial x_l} = \frac{\partial \text{frelu}(x_l)}{\partial x_l} \frac{\partial \text{Cost}}{\partial \hat{x}_l}$. According to Equ. (5), we know that $E \left[ \left( \frac{\partial \text{Cost}}{\partial x_l} \right)_l \right]^2 = \frac{1}{2} \text{Var} \left[ \frac{\partial \text{Cost}}{\partial \hat{x}_l} \right]$. Therefore, $\text{Var} \left[ \frac{\partial \text{Cost}}{\partial x_l} \right] = \frac{1}{2} \hat{n}_l \text{Var}[w_l] \text{Var} \left[ \frac{\partial \text{Cost}}{\partial \hat{x}_l} \right]$. Then for a network with $L$ layers, we have $\text{Var} \left[ \frac{\partial \text{Cost}}{\partial x_1} \right] = V \text{ar} \left[ \frac{\partial \text{Cost}}{\partial x_L} \right] \left( \prod_{l=2}^{L-1} \frac{1}{2} \hat{n}_l \text{Var}[w_l] \right)$. Therefore, we have the initialization condition:

$$\frac{1}{2} \hat{n}_l \text{Var}[w_l] = 1, \forall l,$$  

(6)

which is the same with the msra method [8] for ReLU.

2) Forward propagation: For the forward propagation case, that is $x_l = W_l \hat{x}_l$, where $W_l$ is a $d$-by-$n$ matrix and $n = k^2 c$. As above, we have $\text{Var}[x_l] = n_l \text{Var}[w_l] E[\hat{x}_l^2]$ with the independent assumption. For FReLU, $\hat{x}_l^2 = \max(0, x_l^2 - 1) + \max(0, 2b_l x_l - 1) + b_l^2$. In general, $x$ is finite or has Gaussian shape around zero, then $E[\hat{x}_l^2] \approx \frac{1}{2} \text{Var}[x_l - 1] + b_l^2$. Thus, we have $\text{Var}[x_l] \approx \left( \frac{1}{2} n_l \text{Var}[x_{l-1}] + n_l b_l^2 \right) \text{Var}[w_l]$. And for a network with $L$ layers, $\text{Var}[x_L] = \text{Var}[x_1] \prod_{l=2}^{L-1} \frac{1}{2} n_l \text{Var}[w_l] + \xi$, where $\xi = \sum_{k=2}^{L} \left( b_l^2 \frac{1}{2} n_l \text{Var}[w_l] \right)$. We found that the term $\xi$ makes forward propagation more complex. Fortunately when using Equ. (6) for initialization, $\text{Var}[x_L] \approx \frac{1}{2 L} \text{Var}[x_1] + \sum_{k=2}^{L} \frac{c_l}{d_l} b_l^2$.

In conclusion, when using the initialization condition (Equ. (6)) for FReLU, the variance of back propagation is stable and the variance of forward propagation will be scaled by some scalars. FReLU has a relatively stable learning characteristic except in complex applications. Thus, for stable learning, the absolute of $b_l$ prefers to be a small number, especially for very deep models. In practice, by using batch normalization [17], networks will be less sensitive to the initialization method. And the data-driven initialization method LSUV [19] is also a good choice. For convenience, in this paper, we use MSRA method [8] (Equ. (6)) for all our experiments.

C. Analysis and Discussion for FReLU

In this subsection, we analyze and discuss the improvement of FReLU for neural networks.

1) State Extension by FReLU: By adding a learnable bias term, the output range of FReLU is $[b, +\infty)$, which is helpful to ensure efficient learning. When $b < 0$, FReLU satisfies the principle that activation functions with negative values can be used to reduce bias effect [10]. Besides, negative values can improve the expressiveness of the activation function. There are three output states represented by FReLU with $b < 0$:

$$frelu(x) = \begin{cases} 
  \text{positive} & \text{if } x > 0 \text{ and } x + b > 0 \\
  \text{negative} & \text{if } x > 0 \text{ and } x + b < 0 \\
  \text{inactivation} & \text{if } x \leq 0 
\end{cases}.$$  

(7)

Considering a layer with $n$ units, FReLU with $b = 0$ (equal to ReLU) or $b > 0$ can only generate $2^n$ output states, while FReLU with $b < 0$ can generate $3^n$ output states. As shown in Table III, the learnable biases tend to negative and bring the improvement in the network by a successful training. At the same time, FReLU retains the same non-linear and sparse characteristics with ReLU. In addition, the self-adaptation of FReLU is also helpful to find a specialized activation function.
2) Batch Normalization with FReLU: According to the conclusion in [10] and the experiments in Table I, PReLU, SReLU, and ELU are not compatible with batch normalization (BN) [17]. It is because of the training conflict between the representation restoration (scale $\gamma$ and bias $\beta$) in BN and the negative part in the activation function. For FReLU, the operation $\max(x, 0)$ isolates the two parts of learnable terms between BN and FReLU, thus reducing the conflict. In this paper, we introduce BN [17] to stabilize the learning when using a large learning rate for achieving better performance. With BN, backward propagation through a layer is unaffected by the scale of its parameters [17]. BN is also a data-driven method, does not rely on strict distribution assumptions. We show the compatibility between BN and FReLU in our experiments (Table I).

D. Comparisons

We compare the proposed FReLU function with a few related activation functions, including ReLU, PReLU, ELU, and SReLU.

1) ReLU: ReLU [2] is defined as $relu(x) = \max(x, 0)$. FReLU is an extension of ReLU by adding a learnable bias term $b$ (Eq. (3)). Therefore, FReLU retains the same non-linear and sparse properties as ReLU, and extends the output range from $[0, +\infty)$ to $[b, +\infty)$. Here, $b$ is a learnable parameter for adaptive selection by training. When $b = 0$, FReLU generates ReLU. When $b > 0$, FReLU tends to move the output distribution of ReLU to larger positive areas, which is unnecessary for state extension. When $b < 0$, FReLU expands the states of the output to increase the expressiveness of the activation function.

2) PReLU/LReLU: PReLU [8] is defined as $\text{prelu}(x) = \max(x, 0) + k \min(x, 0)$, where $k$ is the learnable parameter. When $k$ is a small fixed value, PReLU becomes LReLU [7]. To avoid zero gradients, PReLU and LReLU propagate the negative input with penalization, thus also avoid negative missing. However, PReLU and LReLU probably lose sparsity, which is an important factor to achieve good performance for neural networks. Note that FReLU also can generate negative outputs, but in a different way. FReLU obstructs the negative input the same as ReLU. The backward gradient of FReLU for the negative input is zero, which retains sparsity.

3) ELU: ELU [10] is defined as $elu(x) = \max(x, 0) + \min(exp(x-1), 0)$. FReLU and ELU have similar shapes and properties in some extent. Different from ELU, FReLU uses the bias term instead of the exponential operation. Although FReLU is non-differentiable at $x = 0$, the experiments show that FReLU can achieve good performance. In addition, FReLU has a better compatibility with BN than ELU.

4) SReLU: In this paper, shifted ReLU (SReLU) is defined as $srelu(x) = \max(x, \Delta)$, where $\Delta$ is the learnable parameter. Both SReLU and FReLU have the flexibility of choosing vertical shifts. Specifically, SReLU can be reformed as $srelu(x) = \max(x, \Delta) = \max(x - \Delta, 0) + \Delta$. To some extent, SReLU is equivalent to FReLU. But in the experiments, we find that SReLU is less compatible with batch normalization and lower performance than FReLU.

III. EXPERIMENTS

In this section, we evaluate FReLU on three standard image classification datasets, including CIFAR-10, CIFAR-100 [20] and ImageNet [21]. We conduct all experiments based on fb.resnet.torch [22] and use the default data augmentation and training settings. The default learning rate is initially set to 0.1. The weight decay is set to 0.0001, and the momentum is set to 0.9. For CIFAR-10 and CIFAR-100, the models are trained by stochastic gradient descent (SGD) with the batch size of 128 for 200 epochs (no warming up). The learning rate is decreased by a factor of 10 at 81 and 122 epochs. For ImageNet, the models are trained by SGD with the batch size of 256 for 90 epochs. The learning rate is decreased by a factor of 10 every 30 epochs. In addition, the parameter $b$ for FReLU is set to $-1$ as the initialization by default in this paper. For fair comparison and reducing the random influences, all experimental results on CIFAR-10 and CIFAR-100 are reported with the mean and standard deviation of five runs with different random seeds.

A. The Analysis for FReLU

1) Convergence Rate and Performance: We first evaluate the proposed FReLU on a small convolutional neural network (referred to as SmallNet). It contains 3 convolutional layers followed by two fully connected layers (detailed in Table II). The activation module (ACT) is either ReLU, PReLU, ELU, SReLU or FReLU. We used the SmallNet to perform object classification on the CIFAR-100 [20] dataset. Both training and test error rates are reported in Table I and we also draw several learning curves in Fig. 3. We find that FReLU achieves fast convergence and competitive generation performance. Note that the error rate on the test set is lower than on the training set is a normal phenomenon for a small network on CIFAR-100.

2) The Effect of Bias in Convolution Layer: In Table I, we report two groups of results. The first and second columns are the results of not using biases in convolution layers. The third column is the results of using biases in convolution layers. We

https://github.com/facebook/fb.resnet.torch

1225
We investigate the compatibilities with batch normalization (BN) on the experiments (Section III-B and Section III-C). We observe that without BN, using biases in convolution layers is important for ELU and SReLU to achieve good performance. FReLU is fine for both cases. And the network with BN generates the same results in both cases. Thus, we do not use biases in convolution layers when using BN in the following experiments (Section III-B and Section III-C).

3) Compatibility with Batch Normalization: We investigate the compatibilities with batch normalization (BN) on the SmallNet. As the same in [10], BN improves ReLU networks but damages ELU networks. We also empirically find that BN does not improve PReLU, SReLU, and FReLU when the base learning rate equals to 0.01. No matter with or without BN, FReLU achieves competitive testing error rates. Moreover, when using large base learning rate 0.1, ReLU, PReLU, ELU, SReLU, and FReLU networks all cannot converge without BN. With higher learning rates, ReLU, PReLU, and FReLU enjoy the benefits of BN, but ELU and SReLU do not. These phenomenon reflect that FReLU is compatible with BN, which avoids exploding and achieves better performance with large learning rate.

4) Different Initialization Values for FReLU: Here we further explore the effects of different initialization values for FReLU. We report the results on the CIFAR-100 dataset with the SmallNet. By using a small network, the parameter of FReLU can be fully learned. The test error rates and the convergence values $b$ are shown in Table III. Interestingly, networks with different initialization values (including positive and negative values) for FReLU are finally converged to the close negative value. As discussed in [23], assuming the input $x \sim N (0, 1)$, the output expectation of activation function $f (x)$ can be expressed as $E [x] = \int_{-\infty}^{\infty} x \exp (-0.5x^2) f (x)$. When the parameter of FReLU $b \approx -0.398$ proven in Table III, $E [x]$ is approximately equal to zero. Therefore, FReLU can be a normalized activation function to ensure the normalization of the entire network.

### Table I
Comparing ReLU [5], PReLU [8], ELU [10], SReLU, and FReLU with SmallNet on the CIFAR-100 dataset. We report the mean (STD) error results over five runs. The first and second columns are the results of not using biases in convolution layers. The third column is the results of using biases in convolution layers.

| Method                | Training/0.01 | Test/0.01 | Training/0.1 | Test/0.1 | Up: 0.01, Bottom: 0.1 |
|----------------------|---------------|-----------|---------------|-----------|------------------------|
| ReLU                 | 44.20 (0.31)  | 40.55 (0.25) | not converge  | not converge | 42.33 (0.23)           |
| PReLU                | 42.49 (0.12)  | 38.48 (0.33) | exploding     | exploded   | 40.70 (0.12)           |
| ELU                  | 40.79 (0.14)  | 37.55 (0.47) | exploding     | exploded   | 39.34 (0.17)           |
| SReLU                | 41.41 (0.08)  | 38.31 (0.08) | exploding     | exploded   | 39.94 (0.10)           |
| FReLU                | 38.69 (0.17)  | 36.87 (0.35) | exploding     | exploded   | 38.40 (0.09)           |
| BN+ReLU              | 44.07 (0.18)  | 39.20 (0.32) | 42.60 (0.16)  | 38.30 (0.43) | 41.40 (0.14)           |
| BN+PReLU             | 42.46 (0.27)  | 39.42 (0.54) | 40.85 (0.17)  | 37.14 (0.42) | 41.04 (0.25)           |
| BN+ELU               | 45.10 (0.18)  | 38.77 (0.18) | 43.27 (0.11)  | 37.80 (0.16) | 42.63 (0.15)           |
| BN+SReLU             | 43.47 (0.09)  | 38.22 (0.28) | 40.15 (0.07)  | 37.20 (0.26) | 40.11 (0.11)           |
| BN+FReLU             | 40.38 (0.26)  | 37.13 (0.30) | 38.83 (0.18)  | 35.82 (0.12) | 38.00 (0.07)           |

### Table II
SmallNet architecture on the CIFAR-100 dataset. (BN: Batch Normalization; ACT: activation function.)

| Type       | Patch Size/Stride | #Kernels |
|------------|-------------------|----------|
| Convolution | 3x3/1             | 32       |
| (BN +) ACT | –                  | –        |
| MAX Pool   | 2x2/2              | –        |
| Dropout (20%) | –              | –        |
| Convolution | 3x3/1             | 64       |
| (BN +) ACT | –                  | –        |
| MAX Pool   | 2x2/2              | –        |
| Dropout (20%) | –              | –        |
| Convolution | 3x3/1             | 128      |
| (BN +) ACT | –                  | –        |
| MAX Pool   | 2x2/2              | –        |
| Dropout (20%) | –              | –        |
| (BN +) ACT | –                  | –        |
| Dropout (50%) | –              | –        |
| Linear     | –                  | 512      |
| Softmax    | –                  | 100      |

Fig. 3. Error curves on the CIFAR-100 dataset for SmallNet. The base learning rate is 0.01 (the first column in Table I). Best viewed in color.
### Table III
**Mean (std) error results on the CIFAR-100 dataset and convergence values (Layer 1 to 4) for FReLU with SmallNet.**

| Init. Value | Error Rate | Layer1 | Layer2 | Layer3 | Layer4 |
|-------------|------------|--------|--------|--------|--------|
| 0.5         | 37.05 (0.07) | -0.3175 | -0.4570 | -0.2824 | -0.3284 |
| 0.2         | 36.71 (0.32)  | -0.3112 | -0.4574 | -0.2749 | -0.3314 |
| 0           | 36.91 (0.34)  | -0.3144 | -0.4367 | -0.2891 | -0.3313 |
| -0.4        | 37.10 (0.33)  | -0.3235 | -0.4480 | -0.2917 | -0.3315 |
| -1          | 36.87 (0.35)  | -0.3272 | -0.4757 | -0.2849 | -0.3282 |

Fig. 4. The distribution of deeply learned features for (a) ReLU and (b) FReLU on the test set of MNIST dataset. The points with different colors denote features from different classes. Best viewed in color.

5) **Visualize the Expressiveness of FReLU:** Here we further visualize the deep feature embeddings for ReLU and FReLU. We conduct this experiment on the MNIST [24] dataset with LeNets++. As the output dimension of the last hidden layer in LeNets++ is 2, we can directly plot features on the 2-D surface for visualization. In LeNets++, we use ReLU as the activation function. To visualize the effect of FReLU for feature learning, we replace the activation function of the last hidden layer as FReLU. We draw the embeddings on the test dataset after training in Fig. 4. We observe that embeddings of the FReLU network are more discriminative than ReLU’s. Here the parameter $b$ of FReLU is around $-6$. The accuracy of the FReLU network is 97.8%, while the ReLU network is 97.05%. With the negative bias, FReLU provides larger space for feature representation than ReLU.

### B. Results on CIFAR-10 and CIFAR-100

1) **Results on Network in Network:** In this subsection, we compare ReLU, PReLU, ELU, SReLU and FReLU on the Network in Network (referred to as NIN) [25] model. We report results on both CIFAR-10 and CIFAR-100 datasets in Table IV. We use the default base learning rate 0.1 and test with BN. PReLU seems overfitting and does not obtain good performance. FReLU achieves the competitive performance.

2) **Evaluation on Residual Networks:** Results of FReLU with residual networks are shown in Table V. To compare the compatibility of FReLU and ELU with BN, we first investigate the performances with simply replacing the ReLU with FReLU and ELU, that is using the architecture in Fig. 5(a). We observe that ELU damages the performance but FReLU improves, which demonstrates that FReLU has the higher compatibility with BN than ELU. Inspired by [26], we further compare the performances with the modified networks, where ELU uses the architecture in Fig. 5(c) and FReLU uses the architecture in Fig. 5(b). We also observe that FReLU achieves better performance.

### C. Results on ImageNet

We also evaluate FReLU on the ImageNet dataset. Table VI shows the results with NIN model and a modified CaffeNet, where the results of CaffeNet comes from a benchmark testing [27] and the detailed settings can refer to their project website. FReLU performs well, outperforming other activation functions.

### IV. Conclusion and Future Work

In this paper, a novel activation function called FReLU is proposed to improve convolutional neural networks. As a variant of ReLU, FReLU retains non-linear and sparsity as ReLU and extends the expressiveness. FReLU is a general concept and does not depend on any specific assumption. We show that FReLU achieves competitive performance and empirically find that it is more compatible with BN than ELU. Our results suggest that negative values are useful for neural networks. There are still many questions requiring further study.

https://github.com/ducha-aiki/caffenet-benchmark/blob/master/Activations.md
TABLE V
Comparing ReLU, ELU ((A) [10] (C) [26]) and FReLU with ResNet-20/32/44/56/110 [4] on the CIFAR-10 and CIFAR-100 datasets. We report the mean (STD) error rates over five runs.

| Dataset | CIFAR-10 | CIFAR-100 |
|---------|----------|-----------|
| #Depths | 20 32 44 | 56 110 |
| Original | 8.12(0.18) | 7.28(0.19) | 6.97(0.14) | 6.87(0.54) | 6.82(0.63) |
| ELU (a) | 8.04(0.08) | 7.62(0.21) | 7.51(0.22) | 7.71(0.26) | 8.21(0.21) |
| FRELU (a) | 8.10(0.18) | 7.30(0.17) | 6.91(0.25) | 6.54(0.22) | 6.20(0.23) |
| ELU (c) | 8.28(0.09) | 7.07(0.17) | 6.78(0.10) | 6.54(0.20) | 5.86(0.14) |
| FRELU (b) | 8.80(0.14) | 6.99(0.11) | 6.58(0.19) | 6.31(0.20) | 5.71(0.19) |
| BN+ReLU | 31.93(0.13) | 30.16(0.32) | 29.30(0.45) | 29.19(0.61) | 28.48(0.83) |
| BN+ELU | 31.90(0.36) | 30.39(0.37) | 29.34(0.39) | 28.81(0.42) | 27.02(0.32) |
| BN+FReLU | 31.84(0.30) | 29.95(0.27) | 29.02(0.25) | 28.07(0.47) | 26.70(0.38) |

Table VI
Comparing ReLU, ELU and FReLU with NIN model on the ImageNet dataset.

| Network | Method | Top-1 error | Top-5 error |
|---------|--------|-------------|-------------|
| NIN     | BN+ReLU | 33.65       | 14.35       |
|         | BN+ELU  | 38.55       | 16.62       |
|         | BN+FReLU| 34.82       | 14.00       |
| CaffeNet| ReLU   | 53.00       | –           |
|         | PReLU   | 52.20       | –           |
|         | ELU     | 51.20       | –           |
|         | FReLU   | 51.20       | –           |

Acknowledgment

This work is supported by Natural Science Foundation of China (No.61702192, No.61751202, and U1636218) and Fundamental Research Funds of Central Universities (2017MS045).

References

[1] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.
[3] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in Proc. 31th AAAI Conf. Artif. Intell., 2017.
[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Computer Vision Pattern Recognition, 2016, pp. 770–778.
[5] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in Proc. 27th Int. Conf. Mach. Learn., 2010, pp. 807–814.
[6] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in Proc. 14th Int. Conf. Artif. Intell. Stat., vol. 15, 2011, pp. 315–323.
[7] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in Proc. 30th Int. Conf. Mach. Learn. Workshop, vol. 28, 2013.
[8] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in Proc. IEEE Int. Conf. Comput. Vision, 2015, pp. 1026–1034.
[9] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” CoRR, vol. abs/1505.00853, 2015.
[10] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” ICLR, 2016.
[11] L. Trottier, P. Giguère, and B. Chaib-draa, “Parametric exponential linear unit for deep convolutional neural networks,” 16th IEEE Int. Conf. Mach. Learn. Appl., pp. 207–214, 2016.
[12] Y. Li, C. Fan, Y. Li, and Q. Wu, “Improving deep neural network with multiple parametric exponential linear units,” CoRR, vol. abs/1606.00305, 2016.
[13] B. Carlile, G. Delamarre, P. Kinney, A. Marti, and B. Whitney, “Improving deep learning by inverse square root linears (isrls),” CoRR, vol. abs/1710.09967, 2017.
[14] G. Klimbauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-normalizing neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 971–980.
[15] R. Duggal and A. Gupta, “P-telu: Parametric tan hyperbolic linear unit activation for deep neural networks,” in Proc. IEEE Conf. Computer Vision Pattern Recognition, 2017, pp. 974–978.
[16] B. Xu, R. Huang, and M. Li, “Revise saturated activation functions,” CoRR, vol. abs/1602.05980, 2016.
[17] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” vol. 37, 2015, pp. 44–486.
[18] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in Proc. 13th Int. Conf. Artif. Intell. Stat., 2010, pp. 249–256.
[19] D. Mishkin and J. Matas, “All you need is a good init,” CoRR, vol. abs/1101.06422, 2015.
[20] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.
[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., “Imagenet large scale visual recognition challenge,” Int. J. Computer Vision, vol. 115, no. 3, pp. 211–252, 2015.
[22] S. Gross and M. Wilber, “Training and investigating residual nets,” Facebook AI Research, CA.[Online]. Available: http://torch.ch/blog/2016/02/04/resnets. html, 2016.
[23] D. Arpit, Y. Zhou, B. Kato, and V. Govindaraju, “Normalization propagation: A parametric technique for removing internal covariate shift in deep networks,” in Proc. 33rd Int. Conf. Mach. Learn., vol. 48, 2016, pp. 1168–1176.
[24] C. J. B. Yann LeCun, Corinna Cortes, “The mnist database of handwritten digits,” http://yann.lecun.com/exdb/mnist/, 1998.
[25] M. Lin, Q. Chen, and S. Yan, “Network in network,” Computer Science, 2013.
[26] D. Mishkin, N. Sergeevsky, and J. Matas, “Systematic evaluation of convolution neural network advances on the imagenet,” Computer Vision and Image Understanding, vol. 161, pp. 11–19, 2017.