Neurocomputing 000 (2017) 1–9

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Adaptive activation functions in convolutional neural networks

Sheng Qian[a], Hua Liu[b], Cheng Liu[a], Si Wu[b], Hau San Wong[a,*]

[a] Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong, PR China
[b] Department of Computer Science, South China University of Technology, Guangzhou, Guangdong, PR China

ARTICLE INFO

Article history:
Received 22 December 2016
Revised 1 May 2017
Accepted 15 June 2017
Available online xxx

Communicated by Prof. Yicong Zhou

MSC:
00-01
99-00

Keywords:
Convolutional neural networks
Activation function learning
Adaptive activation
Hierarchical activation

ABSTRACT

Activation functions play important roles in deep convolutional neural networks. This work focuses on learning activation functions via combining basic activation functions in a data-driven way. We explore three strategies to learn the activation functions, and allow the activation operation to be adaptive to inputs. We firstly explore two strategies to linearly and nonlinearly combine basic activation functions, respectively. Then we further investigate a strategy that basic activation functions are combined in a way of a hierarchical integration. Experiments demonstrate that the proposed activation functions lead to better performances than ReLU and its variants on benchmarks with various scales.

1. Introduction

These years have witnessed the remarkable development of CNNs in various computer vision applications [1], such as image classification [2,3], image retrieval [4], object detection [5] and tracking [6,7]. Non-saturated activation functions rather than saturated versions play key roles in current deep models, which have promoted the success of deep models. Compared with saturated versions, they are beneficial to deep model learning. On the one hand, they help to tackle the problem that gradients will tend to vanish when using saturated versions. On the other hand, they contribute to accelerating model learning.

Considerable attention has been paid to the research of non-saturated activation functions during the past years. The rectified linear unit (ReLU) [8] has been generally applied in these works [2,9–11]. This is a piecewise linear function whose positive part is an identity function and negative part is set to zero. Compared with ReLU, the leaky ReLU (LReLU) [12] assigns a relative smaller and predefined slope to the negative part. Furthermore, the randomized ReLU (RReLU) [13] samples the slope from a uniform distribution during training and sets it to the mean of the distribution during test, which helps to decrease the overfitting via this randomized strategy. Through generalizing LReLU, the parametric ReLU (PReLU) [14] allows the slope to be learned from data and improves the representation ability. In contrast with the linearity of negative parts in above activation functions, the exponential linear unit (ELU) [15] exponentially reduces the slope from a predefined and fixed threshold to zero, and is beneficial to speed up model learning. Inspired by PReLU, we also propose a parametric variant of ELU called parametric ELU (PELU).

In addition, more complex forms of activation functions have been proposed to improve the ability of learning non-linear transformation. Considering the objective of leveraging the dropout model averaging technique in models design, the maxout unit (Maxout) [16] computes the maximum of linear units in group to approximate convex activation functions. With keeping the desirable property of the maxout unit, the probabilistic Maxout (Probout) [17] uses a probabilistic sampling procedure to select the maximum of linear units to take full advantage of the linear subspace composed of these linear units. By learning basic rectified linear units, the adaptive piecewise linear activation (APL) [18] can approximate both convex and nonconvex functions composed of a set of generalized hinge-shared linear units.

In this paper we are interested in ReLU and its variants, namely the rectified unit family. According to different negative parts of these activation functions, we categorize them into three types: zero-type, linear-type and exponential-type. Among them, ReLU

* Corresponding author.
E-mail addresses: sqian9-c@my.cityu.edu.hk (S. Qian), andyliu9309@gmail.com (H. Liu), cliu272-c@my.cityu.edu.hk (C. Liu), cswusi@scut.edu.cn (S. Wu), cshswong@cityu.edu.hk (H.S. Wong).

http://dx.doi.org/10.1016/j.neucom.2017.06.070
0925-2312/

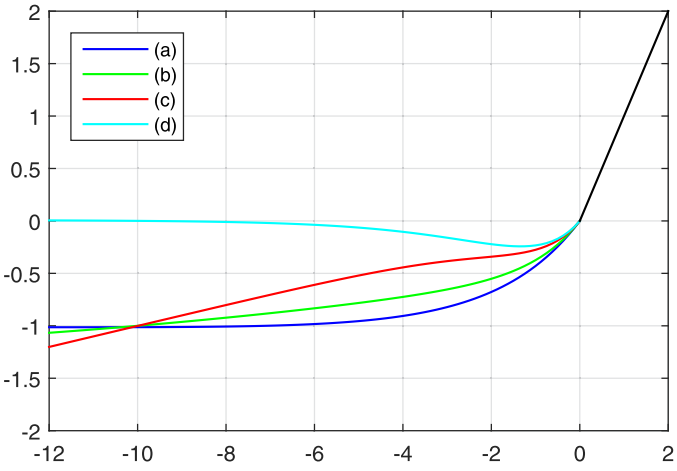

Fig. 1. Activation functions of middle-level nodes in hierarchical activation with different learned parameters. These activation functions have various forms. Among them, (a), (b) and (c), (d) are convex and non-convex functions, respectively. All activation functions share the positive parts indicated by the black line.

belong to the first type, LReLU and PReLU belong to the second type, and ELU and PELU belong to the third type. The zero-type can be viewed as a special form of both the linear-type and the exponential-type. Their detailed relationship will be pursued further in Section 2. The rectified unit family has been widely applied in the recent popular deep CNNs [11,14,15,19,20]. When choosing activation functions for a specific CNN, activation functions in all activation layers in this CNN are confined to only one of the above types. Since their functional forms are designed as convex functions, it has restricted the representation ability of learning nonlinear transformation to some degree. By utilizing this rectified unit family, we aim to design an activation function to improve the ability of learning non-linear transformation and be adaptive to the inputs. We expect the designed activation function to have more flexible forms which can be determined in a data-driven way.

First, we adopt two strategies to combine basic activation functions. The one is denoted as mixed activation, in which the activation operation is learned by linearly combining basic activation functions. In order to make the learned activation operation be adapted to the specific inputs, the other is gated activation, in which the activation operation is learned by nonlinearly combining basic activation functions. For the sake of further improving the ability of learning non-linear transformation, we extend the above structures and propose the hierarchical activation as the third strategy. The combination of basic activation functions will be learned and organized as a unit in a more complex hierarchical structure. The goal of hierarchical activation is to allow basic activation functions being combined to be learned directly from the data and be adapted to the specific inputs. For this purpose, the winner-take-all mechanism is adopted to enhance the nonlinearity of the activation operation. Specifically, we design a three-level hierarchical structure. The low-level nodes are associated with basic activation operations. Each middle-level node is associated with the combination of a pair of low-level nodes and the high-level node corresponds to the entire output generated through integrating all of middle-level nodes. As a result, the hierarchical activation can be considered as an integrated scheme having the adaptability to the inputs. Fig. 1 shows various function forms of middle-level nodes in hierarchical activation and Fig. 5 illustrates the basic structure of hierarchical activation. More details will be described in Section 3. To verify the effectiveness of the proposed strategies, we perform the validation experiments with multiple deep CNNs on multiple datasets.

The most relevant literature of this work is generalizing pooling functions (GPF) [21]. GPF explores various strategies (mixed, gated, and tree) to learn pooling operations in CNNs. Compared with GPF, this work focuses on learning activation operations. In addition, we list the main differences between these two works as follows. The tree strategy in GPF adopts a complete binary tree to organize its structure. It simply extends the gated strategy and implicitly increases the depth of the whole network. A deeper network is actually more difficult to optimize. Different from GPF, the hierarchical strategy in this work uses the forest composed of multiple binary trees of depth 1 to organize its structure. It considers the characteristics of activation functions, and introduces the winner-take-all mechanism [22,23] which is beneficial for improving the nonlinearity of activation functions. Meanwhile it does not implicitly increase the depth of the whole network.

The main contribution of this work includes: we propose the mixed activation and the gated activation to linearly and nonlinearly combine basic activation functions in a data-driven way, respectively. As an extension of gated activation, hierarchical activation further improves the ability of learning non-linear transformation, and is equipped with the ability of being adaptive to the inputs. In multiple deep CNNs investigated, such as NIN [24], All-CNN [25], ELUNet [15], replacing standard activation functions with the proposed activation functions achieves a performance boost on benchmarks with various scales including MNIST [26], CIFAR [27], and ImageNet [28].

2. Rectified unit family

In the rectified unit family the zero-type can be viewed as a special form of both the linear-type and exponential-type. Hence, we just need to discuss two types of activation functions: linear-type and exponential-type. Formally, we define LReLU and ELU defined as:

$$f_{\text{lrelu}}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \le 0, \end{cases} \quad f_{\text{elu}}(x) = \begin{cases} x & \text{if } x > 0 \\ \beta(\exp(x) - 1) & \text{if } x \le 0 \end{cases} \tag{1}$$

where x is the input of activation functions $f_{\text{lrelu}}(\cdot)$ and $f_{\text{elu}}(\cdot)$. α and β are weights controlling the slope of the negative part. Both α and β are predefined hyper-parameters. These activation functions belong to the rectified unit family and their relationship can be observed from the configuration of α and β. When $\alpha = 0$ and $\beta = 0$, both LReLU and ELU become ReLU. If α and β become learnable parameters, then LReLU and ELU will become PReLU and PELU respectively. For PReLU and PELU, we simply denote their formulas by $f_{\text{prelu}}(\cdot)$ and $f_{\text{pelu}}(\cdot)$. We can compute the error signal to be propagated back to the previous layer and the gradients with respect to the parameters α and β:

$$\frac{\partial f_{\text{prelu}}(x)}{x} = \begin{cases} 1 & \text{if } x > 0 \\ \alpha & \text{if } x \le 0, \end{cases} \quad \frac{\partial f_{\text{prelu}}(x)}{\alpha} = \begin{cases} 0 & \text{if } x > 0 \\ x & \text{if } x \le 0 \end{cases} \tag{2}$$

$$\frac{\partial f_{\text{pelu}}}{x} = \begin{cases} x & \text{if } x > 0 \\ \beta \exp(x) & \text{if } x \le 0, \end{cases}$$

$$\frac{\partial f_{\text{pelu}}(x)}{\beta} = \begin{cases} 0 & \text{if } x > 0 \\ \exp(x) - 1 & \text{if } x \le 0 \end{cases} \tag{3}$$

3. Adaptive activation functions

Convolutional neural networks are commonly structured as directed acyclic graphs whose vertexes are convolutional layers, activation layers and pooling layers. Our aim is to introduce learning and adaptation into the activation operation. We begin with activation functions with predefined parameters (LReLU and ELU), and

Please cite this article as: S. Qian et al., Adaptive activation functions in convolutional neural networks, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.06.070