Spike buffer: improve deep network performance by offset mechanism

Daihui Li†, ‡, Shangyou Zeng†, Chengxu Ma†
†College of Electronic Engineering, Guangxi Normal University, Guilin, Guangxi, People’s Republic of China
‡E-mail: ldhcv@sina.com

Abstract: For a well-designed neural network model, it is difficult to further improve its performance. This study proposes an offset mechanism called spike buffer, which can effectively improve the performance of the designed convolutional neural networks. The spike buffer introduces an offset buffer-bit and a gradient spike function in the convolution channels to enhance the expression of effective features and suppresses the extraction of invalid features. Without significantly increasing the computational complexity of deep convolution neural networks, it can improve the feature selection performance of convolution neural networks and enhance the ability of non-linear mapping, and can be easily embedded into various convolution neural networks. Experiments show that the performance of convolutional neural networks with integrated spike buffer can be effectively improved.

1 Introduction

Deepening the convolutional network, widening the width of the convolutional neural network and completely changing the network structure for intelligent systems with convolutional neural networks may improve neural network performance to some extent. However, destroying the original convolutional neural network structure may lead to changes in the receptive field, which is obtained by the convolutional layers. It will destroy the original design of the intelligent system, which will eventually affect the performance of the entire intelligent system. Moreover, deepening the convolutional network and broadening the convolutional neural network will lead to excess computation. There are several training tricks that can improve the performance of intelligent systems, not just from structural improvements [1, 2]. Therefore, we have devised a new way to further improve the performance of intelligent systems with a fixed convolutional neural network structure. The proposed offset mechanism is called a spike buffer (SB). It can be used in conjunction with many tricks that do not rely on structural improvements. SB can be easily embedded in various convolutional neural networks that will be trained or pre-trained, which can effectively improve the performance of convolutional neural networks.

The classical convolutional neural networks perform feature filtering through convolutional layers and perform non-linear transformation through activation functions. The performance of feature extraction with a single convolutional layer depends on the size and numbers of convolution kernels. The random initialisation of convolution kernel parameters and the training mode of random data input lead to the inexplicability of features. It also leads to an uneven distribution of efficient convolution kernels and inefficient convolution kernels. The non-linear characteristics affect the fitting ability of the deep networks. Traditional convolutional neural networks add a non-linear mapping to the active layers after the convolutional layers. Based on the learning mechanism of the neural networks, an effective buffer-bit is established in the convolutional layer and activated by the gradient spike function. It promotes the enhancement of high-performance convolution kernels and the suppression of low-performance convolution kernels by convolutional neural networks. SB introduces strong non-linearity and improves the expressiveness of the networks without significantly increasing the computational expense.

2 Spike buffer

We focus on the backpropagation mechanism of neural networks, and introduce buffer-bit and gradient spike function into neural networks to form SB. When the convolutional neural network propagates forward, the SB adapts the weights to different feature maps according to the importance of the feature map. When the convolutional neural network is back-propagating, the SB adapts the gradient of the convolution kernel according to the efficiency of convolution kernels (Fig. 1).

2.1 Buffer-bit

The convolutional layer of the convolutional neural networks generates new feature maps by performing a convolution operation on the current feature maps by the convolution kernels

\[
\begin{align*}
x_1' &= \text{Conv}(x_{1-1}', w_j') + b_j' \\
x_2' &= \text{Conv}(x_{2-1}', w_j') + b_j' \\
&\vdots \\
x_l' &= \text{Conv}(x_{l-1}', w_j') + b_j' \\
z_l' &= f(x_1', x_2', \ldots, x_l')
\end{align*}
\]  

(1)

where \( j \) is the serial number of current convolutional layers. The number of convolution kernels is set to \( i \), and the output of the
conventional layers with serial number \( j \) is denoted by \( z^j \). The weight matrix of the first convolution kernel of the convolution layers with serial number \( j \) is denoted by \( w \). The output of the first convolution kernel convolves the previous feature maps of the convolution layers with serial number \( j \), meanwhile we name the output \( x^j \). The bias of the convolution layers with serial number \( j \) is denoted by \( b^j \). Conv(\( \ast \)) is the convolution operation and \( f(\ast) \) is the activation function of current convolutional layers. The unit occupancy space after the per-convolution channel is called buffer-bit. Buffer-bit is a unit consisting of a single value, and the buffer-bits with gradient spike function are called spike buffers

\[
\begin{align*}
\frac{\partial E}{\partial w_i} &= \frac{\partial E}{\partial (Bu \cdot \text{Conv})} \frac{\partial (Bu \cdot \text{Conv})}{\partial w_i} \\
\frac{\partial E}{\partial u_i} &= \frac{\partial E}{\partial (Bu \cdot \text{Conv})} \frac{\partial (Bu \cdot \text{Conv})}{\partial u_i}
\end{align*}
\]  

where \( Bu \) is the SB with \( i \) buffer-bits, and \( g(\ast) \) is the gradient spike function.

2.2 Gradient spike function

Gradient descent is one of the commonly used methods for unconstrained optimisation problems. When solving the optimisation problems of the loss function, it can be iteratively solved by the gradient descent method to obtain the weight value of the models

\[
\begin{align*}
x^l &= \text{Conv}(z^{l-1}, w^l) \times Bu + b^l \\
Bu &= g(a(u))
\end{align*}
\]

where \( Bu \) is the SB with \( i \) buffer-bits, and \( g(\ast) \) is the gradient spike function.

In order to enhance effective features, suppress invalid features, and further increase the sparsity of convolutional neural networks, we specially design gradient spike function. For forward propagation, \( Bu \) changes its value according to the importance of the features as much as possible that \( Bu \) becomes the weight of the features. In order to achieve this goal, when the neural network uses gradient descent to learn, we can accelerate the descent of the weights with greater gradients and maintain the weights with smaller gradients. Therefore, the gradient of \( a(u) \) in SB becomes particularly important. We need to design an activation function \( g(u) \) that makes \( a(u) \) sensitive to gradient changes. We call \( A(u) \) gradient spike function

\[
\begin{align*}
A(u) &= \frac{pe^{-qv-aq} + 1}{1 + e^{-qv-aq}} + 1
\end{align*}
\]

where parameter 'p' is a spike multiple, which can represent the action intensity of spike function. The parameter 'c' is an intermediate variable inside the SB. And parameter 'a' is the central base point, which is used to control the translation of the spike. The width-factor of the spike is 'd', which can indicate the recovery position of the spike relative to the base point and effectively control the impact range of the spike. The transition factor from spike function to linear function is 'r', which represents the positive offset of gradient relative to 1 when the spike is recovered. In order not to change the initial state of the convolutional neural networks, we initialize \( Bu \) based on Cauchy integral theorem and formula:

\[
x^l = \text{Conv}(z^{l-1}, w^l) \times Bu_{a=1} + b^l
\]

Therefore, it is easy to find the activation function \( g(u) \) of the SB

\[
g(u) = \int A(u)da = \frac{p}{c} \frac{1}{1 + e^{(f-\alpha) \cdot \alpha}} + u - \frac{p}{2c}
\]

Four parameters 'p', 'a', 'r' and 'd' of SB can be adjusted according to the actual use to achieve better performance. We operate on the anomaly of SB (ASB) to make the performance of SB more stable (Fig. 2)

\[
g_{\text{out}}(u) = g(u) - g(\bar{u}) + a
\]

where \( g(\bar{u}) \) is the mean of \( g(u) \), and \( g_{\text{out}}(u) \) is an anomaly.

2.3 Balanced buffer structure (BBS)

In order to easily apply the SB to various networks, this paper designs a BBS. BBS can introduce SB in the form of offset

\[
\text{Conv}_{\text{out}} = \text{Conv}_{\text{in}} + \text{Conv}_{\text{out}} \times Bu_{a=-1}
\]

where \( \text{Conv}_{\text{out}} \) is the output of the current convolutional layers, and \( Bu_{a=-1} \) configures the parameter 'a' of the SB to −1. The BBS can return the initial state of the convolutional layers to zero and automatically adjust the weight of the participating operations according to the computational requirements.

3 Experimental results and discussion

We integrated SB on ResNet [3] and DenseNet [4] for experiments. ResNet and DenseNet represent the best network designed by human beings. And the classic CIFAR10 [5] benchmark dataset was used. The CIFAR10 dataset consists of coloured natural images with 32 × 32 pixels. In the absence of special instructions, parameter 'q' was set to 1, parameter 'd' was set to 0.5 and parameter 'r' was set to 0.1. We first did not use data enhancement to evaluate the impact of performance with different SB parameter configurations on ResNet-18. We used batch size 512 for 60 epochs to train. The initial learning rate was set to 0.1 and the weight decay was set to 0.0025. When the training epoch reached 66.6 and 83.3% of the total training epochs, the learning rate was reduced to one-tenth of the original and weight decay was increased by 0.001 at the same time. The network was trained by using stochastic gradient descent (SGD) as the optimiser.

As can be seen from Table 1, the network with offset mechanism has higher Top-1 accuracy than the baseline network. Top-1 accuracy is the accuracy in the traditional sense, which is the most important indicator of performance. Top-5 accuracy is the correct answer among the five predicted maximum probabilities of the model. The increase in performance caused by different parameter configurations was different. In order to evaluate the performance of the offset mechanism on a deeper network, we experimented with the network that integrated BBS at the residual junction and set parameter 'p' to 12. To further evaluate the applicability of the offset mechanism, we performed experiments under different parameter conditions. We used batch size 512 for 200 epochs to train. The initial learning rate was 1 × 10⁻³, which reduced the learning rate to 1 × 10⁻⁴, 1 × 10⁻⁵, 1 × 10⁻⁶ and 5 × 10⁻¹³ at 80, 120, 160 and 180 epochs, respectively. The network was trained by using Adam as an optimiser and the weight decay was set to 0.0001.

ResNet addressed the degradation problem by introducing a residual structure. Table 2 shows that deeper networks have higher performance. The integration of BBS at the residual junction not only did not bring the gradient to disappear, but also brought performance improvement. Next, we integrated the BBS on DenseNet. Specifically, we integrated BBS in front of each contact
layer of DenseNet. We used batch size 64 for 200 epochs to train. The initial learning rate was 0.1, which reduced the learning rate to 0.01, 0.001 at 100 and 150 epochs. The network was trained by using SGD as an optimiser and the weight decay was set to 0.0001. The experimental results are shown in Table 3.

In the case of default parameter configurations, the Top-1 accuracy of DenseNet-40 with integrated BBS was already 0.13% higher than DenseNet-40 without integrated BBS.

In order to evaluate the amount of computation that SB brought to the convolutional neural network, we counted the training time of ResNet with integrated BBS and ResNet without integrated BBS. Table 4 lists the training time for these networks.

BBS will increase the training time of ResNet by about 10%. And it cannot significantly increase the computing time. Most of the increase in training time may be due to the balanced structure of the BBS. The balanced structure of the BBS is offset by two residual structures. And the balanced structure has one more

---

**Table 1**

| Network | $P$ | Top-1 Acc, % | Top-5 Acc, % |
|---------|-----|--------------|--------------|
| baseline | — | 84.79 | 99.38 |
| SB | 10 | 85.61 | 99.35 |
| | 11 | 86.11 | 99.21 |
| | 12 | 85.97 | 99.41 |
| ASB | 10 | 85.89 | 99.30 |
| | 11 | 85.79 | 99.17 |
| BBS | 10 | 85.91 | 99.47 |
| | 12 | 86.29 | 99.44 |

**Table 2**

| Network | Baseline | $p = 12$ |
|---------|----------|----------|
| ResNet-20 | 89.05 | 89.60 |
| ResNet-32 | 89.84 | 90.11 |
| ResNet-44 | 90.53 | 91.23 |
| ResNet-56 | 91.02 | 91.69 |
| ResNet-101 | 91.44 | 92.30 |

**Table 3**

| Network | Baseline | $p = 12$ |
|---------|----------|----------|
| DenseNet-40 | 93.60 | 93.73 |

**Table 4**

| Network | Baseline | $p = 12$ |
|---------|----------|----------|
| ResNet-20 | 12.3 | 13.3 |
| ResNet-32 | 18.7 | 20.3 |
| ResNet-44 | 25.2 | 27.9 |
| ResNet-56 | 32.3 | 35.1 |
| ResNet-101 | 62.0 | 67.9 |
residual connection than the original, which is the main reason for the increase in the amount of computation.

4 Conclusion

SB is a completely new offset mechanism. Based on the principle of back-propagation, SB gives neural networks the ability to select neurons. It is equivalent to adaptively adjusting the local learning rate to enhance the expression of effective features. And further increase the sparsity of networks. In this paper, various networks of integrated SB are evaluated under various training configurations, and the results prove the validity and applicability of SB. The presence of SB is especially important for performance boosts of fixed-structure neural networks. This paper only proposes a feasible idea, and there are still many problems that require more researches to explore. For example, how to configure the parameters more specifically can make the performance of the convolutional neural network better, what kind of offset mechanism is needed for different neural networks, and even whether the neural network can use a better learning mechanism to train. In addition, SB can improve the performance of neural networks with fixed structures. In theory, SB can be transplanted to various non-convolution neural networks, which is the goal of the next work. We will verify the feasibility of the hypothesis in future work.

5 Acknowledgments

This work was supported by the National Natural Science Foundation of China (grant no. 11465004). Daihui Li, Shangyou Zeng and Chengyu Ma (College of Electronic Engineering, Guangxi Normal University, Guilin 541004, Guangxi, People’s Republic of China).

6 References

[1] He, T., Zhang, Z., Zhang, H., et al.: ‘Bag of tricks for image classification with convolutional neural networks’. The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 558–567
[2] Jia, X., Song, S., He, W., et al.: ‘Highly scalable deep learning training system with mixed-precision: training imagenet in four minutes’, https://arxiv.org/abs/1807.11205, July 2018
[3] He, K., Zhang, X., Ren, S., et al.: ‘Deep residual learning for image recognition’. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770–778
[4] Huang, G., Liu, Z., van der Maaten, L., et al.: ‘Densely connected convolutional networks’. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Hawaii, USA, 2017, pp. 2261–2269
[5] Krizhevsky, A., Hinton, G.: ‘Learning multiple layers of features from tiny images’. Technical Report TR-2009-5, University of Toronto, Toronto