(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
05 November 2020 (05.11.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/220191 A1

(51) International Patent Classification:
G06N 3/08 (2006.01)

(21) International Application Number:
PCT/CN2019/084969

(22) International Filing Date:
29 April 2019 (29.04.2019)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HUAWEI TECHNOLOGIES CO., LTD.** [CN/CN]; Huawei Administration Building, Bantian, Longgang District, Shenzhen, Guangdong 518129 (CN).

(72) Inventors: **ZENG, Tieyong**; The Department Of Mathematics, The Chinese University Of Hong Kong, Shantin, NT, Hong Kong 999077 (CN). **YANG, Hongfei**; The Department Of Mathematics, The Chinese University Of Hong Kong, Shantin, NT, Hong Kong 999077 (CN). **DING, Xiaofeng**; The Department Of Mathematics, The Chinese University Of Hong Kong, Shantin, NT, Hong Kong 999077 (CN). **HU, Hui**; Huawei Administration Building, Bantian, Longgang District, Shenzhen, Guangdong 518129 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) **Title:** METHOD AND APPARATUS FOR TRAINING AND APPLYING A NEURAL NETWORK

(57) **Abstract:** A neural network training method comprises: determining parameters for a weight matrix of each layer of a network, based on the number of shared elements in a structured implementation with shared elements of each layer; determining the weight matrix of each layer based on the determined parameter; decompressing the determined weight matrix to derived an integrated matrix; iterating over forward propagation and backward propagation on all layers of the network based on the integrated matrix until a preset condition is satisfied, wherein the integrated matrix is updated in each iteration; and storing the iterated parameters of the network.

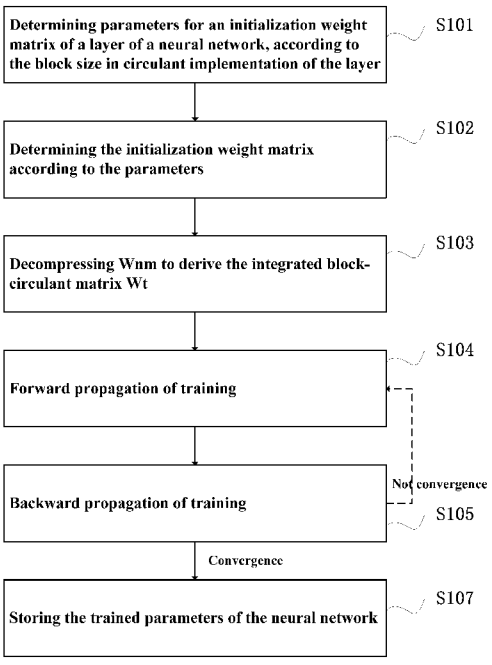

Figure 3

WO 2020/220191 A1

METHOD AND APPARATUS FOR TRAINING AND APPLYING A NEURAL NETWORK

TECHNICAL FIELD

The present disclosure relates generally to artificial intelligence (AI) technique, and more particularly, to a method and apparatus for initialization of compressed neural network with structured matrices with shared elements.

BACKGROUND

In recent years, deep learning methods have remarkable achievements in a broad range of tasks including object classifications, natural language processing, and speech recognition. In 2016, AlphaGo, a Go game software powered by deep learning algorithms, becomes the first Go game software to beat a human champion in a series of 5 Go games. How to effectively train deep neural networks is an active research area. In the seminal work of Xavier and Yoshua, they observed that a proper initialization of the learnable parameters plays an important role in training a neural network. Later, He et al. in extended the method of Xavier and Yoshua to consider the ReLU activation function. Nowadays these two initialization methods are widely used in deep learning software packages like TensorFlow, PyTorch and Keras. Since these two initialization methods only differ by a factor called "gain", which is decided by the activation function, they are considered as one method and called as the Xavier/He initialization.

The main idea of the Xavier/He initialization is to maintain activation variances and back-propagated gradients variance for different layers. However, when there is heavy weigh sharing for certain parameters, the activation variances and the back propagated gradients are not good indicators of variances of learnable parameters. As a result, the update of heavily shared parameters may be much faster than other parameters, and this leads to difficulties in training. If one layer of a neural network is multiplied by a positive constant number and then the positive constant number is divided in another layer, the neural network is very difficult to be trained even if the Xavier/He initialization is used.

SUMMARY

An initialization method is proposed based on the Xavier/He initialization. For a fully connected layer with no weight sharing, the embodiments in the present application are the same as the Xavier/He initialization method. But the embodiments can cope with weight sharing, which is a common phenomenon for CirCNN implementation of neural networks and various numerical embodiments are shown to verify the effectiveness of the initialization method in the present application.

The method is also used to adjust learning rates of parameters layer by layer, which multiplies the weight matrix of each layer by a positive constant scalar, and at the same time change the initialization of this layer accordingly.

In the first embodiment of the present application, a neutral network training method is disclosed, comprising: determining parameters for a weight matrix of each layer of a network, based on the number of shared elements in a structured implementation with shared elements of each layer; determining the weight matrix of each layer based on the determined parameter; decompressing the determined weight matrix to derived an integrated matrix; iterating over forward propagation and backward propagation on all layers of the network based on the integrated matrix until a preset condition is satisfied, wherein the integrated matrix is updated in each iteration; and storing the iterated parameters of the network.

In a feasible implementation, the parameters comprises an adjusting parameter, and wherein the adjusted parameter is negatively correlated with the number of shared elements.

It is noted that the calculation of the adjusting parameter can be 1/B or 1/(B×B) and so on, which is not limited in the present application.

In a feasible implementation, the number of shared elements in the structured implementation with shared elements is a block size in a circulant implementation.

It is noted that circulant implementation is a kind of structured implementation with shared elements, the number of shared elements or the block size relates to a compression ratio of the layer.

In a feasible implementation, the adjusting parameter is calculated as following: $C_{nm} = \frac{1}{\sqrt[4]{B_{nm}}}$, and wherein m is a number of input channel of a layer of the network, n is a number of output channel of the layer of the network, Cnm is the adjusting parameter, and Bnm is the number of shared elements.

In a feasible implementation, the parameters further comprises a random number in an uniform distribution between -Xavier_value and Xavier_value.

It is noted that uniform distribution is a kind of candidate implementation, which is not limited in the present application. Other distributions may be also used for the solution.

In a feasible implementation, Xavier_value is determined by the following:

$$v_{\lambda(n,m)} = \mathrm{Std}\left(v_{\lambda(n,m)}\right) * \sqrt{3.0}$$

where $\mathrm{Std}\left(v_{\lambda(n,m)}\right) = \sqrt{\mathrm{Var}\left(v_{\lambda(n,m)}\right)}$, $\mathrm{Var}\left(v_{\lambda(n,m)}\right) = \mathrm{gain} \cdot \sqrt{B_{nm}} \cdot \frac{2}{m+n}$, gain is a consist number, $v_{\lambda(n,m)}$is the random number.

In a feasible implementation, the weight matrix is determined by the following:

$$W_{nm} = C_{nm} \cdot v_{\lambda(n,m)}$$

where W_{nm} is the weight matrix.

In a feasible implementation, the integrated matrix is updated in each iteration, comprises the random number is updated in each iteration.

In a feasible implementation, the preset condition comprises a training result of the network is convergence.

In a feasible implementation, the training result of the network is convergence, comprises a difference of the weight matrix is smaller than a preset threshold.

It is noted that any criteria which can be used to judge the convergence in a neutral network training process can be applied in the present application.

In a feasible implementation, an input of the network in training comprises image information data or sound information data.

In a feasible implementation, the network is used to classify object, process language or recognize speech.

So it is clear that the present application is useful for the industry, for example, object classifications, natural language processing, and speech recognition field.

In the second embodiment of the present application, a neutral network training device is disclosed, comprising: a first calculation module, to determine parameters for a weight matrix of each layer of a network, based on the number of shared elements in a structured implementation with shared elements of each layer; a second calculation module, to determine the weight matrix of each layer based on the determined parameter; a decompression module, to decompress the determined weight matrix to derived an integrated matrix; an iteration module, to iterate over forward propagation and backward propagation on all layers of the network based on the integrated matrix until a preset condition is satisfied, wherein the integrated matrix is updated in each iteration; and a storage module, to store the iterated parameters of the network.

In a feasible implementation, the parameters comprises an adjusting parameter, and wherein the adjusted parameter is negatively correlated with the number of shared elements.

In a feasible implementation, the number of shared elements in the structured implementation with shared elements is a block size in a circulant implementation.

In a feasible implementation, the adjusting parameter is calculated as following: $C_{nm} = \frac{1}{\sqrt[4]{B_{nm}}}$, and wherein m is a number of input channel of a layer of the network, n is a number of output channel of the layer of the network, Cnm is the adjusting parameter, and Bnm is the number of shared elements.

In a feasible implementation, the parameters further comprises a random number in an uniform distribution between -Xavier_value and Xavier_value.

In a feasible implementation, Xavier_value is determined by the following:

$$v_{\lambda(n,m)} = \mathrm{Std}\left(v_{\lambda(n,m)}\right) * \sqrt{3.0}$$

where $\mathrm{Std}\left(v_{\lambda(n,m)}\right) = \sqrt{\mathrm{Var}\left(v_{\lambda(n,m)}\right)}$, $\mathrm{Var}\left(v_{\lambda(n,m)}\right) = \mathrm{gain} \cdot \sqrt{B_{nm}} \cdot \frac{2}{m+n}$, gain is a consist number, $v_{\lambda(n,m)}$is the random number.

In a feasible implementation, the weight matrix is determined by the following:

$$W_{nm} = C_{nm} \cdot v_{\lambda(n,m)}$$

where W_{nm} is the weight matrix.

In a feasible implementation, the integrated matrix is updated in each iteration, comprises the random number is updated in each iteration.

In a feasible implementation, the preset condition comprises a training result of the network is convergence.

In a feasible implementation, the training result of the network is convergence, comprises a difference of the weight matrix is smaller than a preset threshold.

In a feasible implementation, an input of the network in training comprises image information data or sound information data.

In a feasible implementation, the network is used to classify object, process language or recognize speech.

In the third embodiment of the present application, an apparatus for training a neutral network is disclosed, the apparatus comprising: one or more processors; and a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by the processors, configures the apparatus to carry out the method according to any one of implementation in the first embodiment.

In the fourth embodiment of the present application, a computer program product is disclosed, comprising a program code for performing the method according to any one of implementation in the first embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present disclosure, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, wherein like numbers designate like objects, and in which:

Figure 1 shows the losses of networks summarized in Table 1;

Figure 2 shows the losses of network summarized in formula (12a);

Figure 3 illustrates an exemplary method for a neutral network training;

Figure 4 shows an exemplary compression processing of a block-circulant matrix;

Figure 5 shows an exemplary block diagram of a neutral network training device;

Figure 6 shows an exemplary block diagram of a neutral network training apparatus.

DETAILED DESCRIPTION

Figures 1 through 6, discussed below, and the various embodiments used to describe the principles of the present invention in this patent document are by way of illustration only and should not be construed in any way to limit the scope of the invention. Those skilled in the art will understand that the principles of the invention may be implemented in any type of suitably arranged device or system.

The following documents are hereby incorporated into the present disclosure as if fully set forth herein, the reference numbers of these documents will be used in the following part of this application.

[1] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. Journal of machine learning research, 12(Aug):2493–2537, 2011.

[2] Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, Youwei Zhuo, Chao Wang, Xuehai Qian, Yu Bai, Geng Yuan, et al. Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices. In Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture, pages 395–408. ACM, 2017.

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256, 2010.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In The IEEE International Conference on Computer Vision (ICCV), December 2015.

[6] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al.

Deep neural networks for acoustic modeling in speech recognition. IEEE Signal processing magazine, 29, 2012.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning, pages 448–456, 2015.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[9] Dmytro Mishkin and Jiri Matas. All you need is a good init. In International Conference on Learning Representations, 2016.

[10] Andrew Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. 2013.

[11] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search nature, 529(7587):484, 2016.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for largescale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[13] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In International Conference on Machine Learning, pages 5389–5398, 2018.

[14] Liang Zhao, Siyu Liao, Yanzhi Wang, Zhe Li, Jian Tang, and Bo Yuan. Theoretical properties for neural networks with weight matrices of low displacement rank. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages4082–4090. JMLR. org, 2017.

Some notations and terms used in the present application are introduced here first.

I. Single layer structure

For a typical neural network, one layer of the network is of the form

$$y = act(Wx+b); \quad (1)$$

where $x \in \mathbb{R}^M$ is the input vector, $W \in \mathbb{R}^{N \times M}$ is the weight matrix, $b \in \mathbb{R}^N$ is the bias vector, and act is a pointwise activation function (for example, the identity function or the ReLU function $\gamma \mapsto \max(0, \gamma)$). Entries in W and b are usually parameters to be learned in the training process.

In the training process, a neural network will go through iterations of forward and backward propagations. Suppose the loss function is denoted by L. In a forward propagation, one calculates y based on a given x. In the backward propagation, one calculates $\partial L/\partial W$, $\partial L/\partial x$ and $\partial L/\partial b$ based on a given $\partial L/\partial y$. Suppose that v is a learnable parameter for this layer. Then after one backward propagation with learning rate r, v will be updated by

$$v - r\frac{\partial L}{\partial v} \mapsto v.$$

where $\partial L/\partial v$ will be calculated by using $\partial L/\partial W$, $\partial L/\partial x$ and $\partial L/\partial b$.

For any variable z in this layer, $z^{(0)}$ is used to denote its value after initialization, and $z^{(1)}$ is used to denote its value after the first forward and backward propagation. A definition is introduced here. Assuming the notations and single layer structure as in the reference document [1], let z be a variable in this layer. Then the increment of z, denoted as Δz, is defined to be

$$\Delta z = z^{(1)} - z^{(0)}$$

when we set the learning rate to be 1.

For a learnable variable v, having $\Delta v = -\partial L/\partial v$. For an intermediate variable Wnm which corresponds to the learnable variable v, if it is assumed that v only appears in W in this layer, having

$$\Delta W_{nm} = W_{nm}^{(1)} - W_{nm}^{(0)} = v^{(1)} - v^{(0)} = -\frac{\partial L}{\partial v} = -\sum_{(n',m')} \frac{\partial L}{\partial W_{n'm'}} \quad (2)$$

where the last summation is over all entries in W that share the same learnable parameter v. If v also appears in weights of other layers, the last summation need also to include the corresponding partial derivatives.

II. CirCNN networks

Most state of the art neural networks contain an enormous amount of learnable parameters. For example, the VGG16 network introduced in the reference document [12] contains more than 1 million learnable parameters. How to find efficient ways to decrease the number of parameters in deep neural networks is an active research area. A promising approach is to replace the matrices and convolutions in a deep neural network by structured matrices and structured convolutions. This approach has the advantage that the architecture of the network can be preserved. In reference document [2], it is proposed to replace unstructured matrices in a network by block-circulant matrices, which is called as a CirCNN network. Accordingly, a matrix is a circulant matrix if it has the form

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & a_n \\ a_n & a_1 & \cdots & a_{n-2} & a_{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_2 & a_3 & \cdots & a_n & a_1 \end{bmatrix}$$

Note that unlike an unstructured matrix, a circulant matrix is determined if one knows the first row (or first column) of the matrix. For a fully connected layer of the form y= ReLU(Wx+b), W is a block-circulant matrix if W consists of block matrices, where each block matrix is a circulant matrix. For a fully connected layer, suppose replacing an unstructured matrix by a block-circulant one with block size B. Then for this matrix, reducing the amount of parameters by a factor of B. The number B is called as the compression ratio of this layer. Note that for one layer, the number of parameters in the weight matrix W will dominate the number of parameters in this layer.

It is noted that the unstructured matrix can also be replaced by other structured implementation with shared elements, the compression ratio B relates to the number of shared elements. Circulant implementation is one kind of structured implementations with shared elements. The other structured implementations with shared elements include toplize matrix,

Hankel matrix and so on. For a M×N structured implementations with shared elements, it can be represents by elements fewer than M×N.

Compressing a neural network by using block-circulant matrices can significantly reduce the amount of parameters. For block-circulant matrices, fast algorithms can be employed, like the FFT, to speed up matrix vector multiplication. It has been proved in the reference document [14] that the Universal Approximation Property can be preserved if unstructured matrices are replaced by block-circulant ones, which guarantees the expressive power of CirCNN implementation.

III. The Xavier/He initialization and limitation

An example of a network with a parameter c to show that the Xavier/He initialization may not be sufficient to lead to a fast and stable training.

For a fully connected layer of the form in formula (1), the entries in the matrix W and the vector b are the learnable parameters. Denote the final loss function as L, then the Xavier/He initialization has the following requirements:

$$\begin{cases} \text{mean}(Y) = \text{mean}(X) = 0, \\ \text{var}(Y) = \text{var}(X), \\ \text{mean}(\frac{\partial L}{\partial Y}) = \text{mean}(\frac{\partial L}{\partial X}) = 0, \\ \text{var}(\frac{\partial L}{\partial Y}) = \text{var}(\frac{\partial L}{\partial X}), \end{cases} \tag{3}$$

where it is assumed that all entries in x follow the distribution X, all entries in y follow the distribution Y, all entries in $\frac{\partial L}{\partial x}$ follow the distribution $\frac{\partial L}{\partial X}$, and all entries in $\frac{\partial L}{\partial y}$ follow the distribution $\frac{\partial L}{\partial Y}$. Unless W is a square matrix, the above conditions cannot be satisfied simultaneously, but the following initialization rules are widely used: the bias vector b should be initialized as the zero vector, and entries in W should be identically and independently distributed with mean 0 and variance $\frac{2 \cdot \text{gain}}{M+N}$, where gain is a factor determined by the activation function. For the identity function, gain should be 1, while for the ReLU activation, gain should be 2. In practice

the distributions for entries in W are chosen as either the uniform distribution or the normal distribution.

Considering a network with no bias of the form

$$y = W_2(\mathrm{ReLU}(W_1 x)). \tag{4}$$

where $x \in \mathbb{R}^M$ is the input vector, and W1 and W2 are the parameter matrices of sizes $\mathbb{R}^{M\times M}$ and $\mathbb{R}^{M\times N}$ respectively. Entries in the parameter matrices are to be learned. Decomposing the above model into two networks, each with two layers, in the following two ways

$$\begin{cases} y_1 = \mathrm{ReLU}(W_1 x). \\ y_2 = W_2 y_1. \end{cases} \tag{5a}$$

$$\begin{cases} y_1 = \mathrm{ReLU}(cW_1 x), \\ y_2 = \frac{1}{c} W_2 y_1, \end{cases} \tag{5b}$$

where c is a fixed positive scalar. With the sameW1, W2 and x, these two decompositions will give the same y. By applying Xavier/He initializations to both networks, it can be ensure that for both networks, the mean and variance of the inputs and outputs of each layers are the same, and the mean and variance of the partial derivatives of each layers are also the same. Routine calculations show that initializations should be as follows

$$\begin{cases} \mathrm{var}(W_1) = 2/M. \\ \mathrm{var}(W_2) = 2/(M+N). \end{cases} \tag{6a}$$

$$\begin{cases} \mathrm{var}(W_1) = 2/(c^2 M). \\ \mathrm{var}(W_2) = 2c^2/(M+N). \end{cases} \tag{6b}$$

It can be seen that when c = 1, the two decompositions and initializations are the same. To test the effect of the positive scalar c, the above two networks on the MNIST dataset (a set of handwritten character digits) are tested. In this case M = 784 and N = 10. Setting c = 0:01, and using SGD (Stochastic Gradient Descent, without moment) as the optimizer. The outputs y of each networks

are connected to soft-max layers, and using the cross entropy as the lost function in both cases (for details see the reference document [4]). The learning rates for the two cases are tuned by trial and error to find the best value. It is observed that the training of the second network (with c = 0:01) is much difficult than the training of the first network. As a result, it is needed to use a much smaller learning rate, which results in slow convergence and a higher final loss value. It is clear that for the c = 0:01 case, there is no observable change in the distributions. This indicates that parameters in W1 does not learn after iterations. This is because of the very small learning rate needs to be used and the small value c = 0:01. On the other hand, for the c = 1 case, the distribution evolves from the uniform distribution to a bell-curved distribution. This indicates a successful learning process of parameters in W1. It is noticed that with c = 0:01, the unstable training process cannot be remedied by batch normalization, which is introduced in the reference document [7] and is widely used to remedy unstable training.

A new initialization method for fully connected layers are proposed in the present application.

According to the issues described, a network should be decomposed properly in order to obtain stable training and better network performance. To solve this problem, a new condition to the Xavier/He initialization for layers where the weight matrix does not have sparsity is proposed, which can guarantee a more balanced initialization.

It is assumed that a fully connected layer of a multi-layer network is of the form

$$\begin{cases} W_{nm} = \alpha v_{\lambda(n,m)}, \\ s_n = \sum_m W_{nm} x_m, \\ y_n = \mathrm{act}(s_n + b_n), \end{cases} \tag{7}$$

where α is a positive fixed scalar, $x \in \mathbb{R}^M$ is the input, $y \in \mathbb{R}^N$ is the output, $b \in \mathbb{R}^N$ is the bias, $v \in \mathbb{R}^\Lambda$ is the collection of learnable parameters, act is the activation function, and

$$\lambda : \{0, 1, \cdots, N\} \times \{0, 1, \cdots, M\} \to \{0, 1, \cdots, \Lambda\}$$

is a function that builds the matrix W from the vector v. In this way, it is guarantee that each entry in W corresponds to one element in v, but it is allowed multiple entries in W to share the same

entry in v. Using L to denote the loss function (in the examples in the above description, L is the combination of soft-max and cross-entropy). It is only considered SGD (without moment) as the optimizer. The parameters to be trained are v and b.

Assuming the notations and single layer structure in formula (7). It is require that any initializations of formula (7) should satisfy the following conditions

$$\begin{cases} \text{mean}(Y) = \text{mean}(X) = 0. \\ \text{var}(Y) = \text{var}(X). \\ \text{mean}(\frac{\partial L}{\partial Y}) = \text{mean}(\frac{\partial L}{\partial X}) = 0. \\ \text{var}(\frac{\partial L}{\partial Y}) = \text{var}(\frac{\partial L}{\partial X}), \\ \text{var}(\Delta W) = \text{var}\left(\frac{\partial L}{\partial W}\right). \end{cases} \tag{8}$$

where $\partial L/\partial W$ represents the distribution followed by the entries $\partial L/\partial W_{nm}$.

It is note that for a standard fully connected layer, where all α equals 1 and entries in the matrix W are in one-to-one relations to entries in v, the last condition in formula (8) will be satisfied automatically. The significance of the initialization formula (8) will become apparent when more general fully connected layers with parameter sharing is considered.

For the splitting formula (5b), having $\alpha = c$, and a calculation shows that

$$\text{var}(\Delta W) = \text{var}\left(-c^2\frac{\partial L}{\partial W}\right) = c^4 \cdot \text{var}\left(\frac{\partial L}{\partial W}\right)$$

In order to satisfy the initialization conditions formula (8), c should be equal to 1. For c = 0:01, even though both initializations formula (6a) and (6b) satisfy the Xavier/He initialization, only the initialization formula (6a) satisfies the initialization conditions and it gives a better network decomposition.

In another embodiment, for the CirCNN network in reference document [2], the fully connected layers can be expressed as formula (7) with $\alpha = 1$. A routine calculation shows that

$$\Delta W_{nm} = \Delta v_{\lambda(n,m)} = -\sum_{\lambda(n',m')=\lambda(n,m)} \frac{\partial L}{\partial W_{n'm'}}.$$

denoting $V_{nm} = \{W_{n'm'} : \lambda(n', m') = \lambda(n, m)\}$, and using Bnm to denote the number of elements in Vnm. It is noted that the weight sharing comes from the circulant matrices in W. Assuming that

$$\left\{ \frac{\partial L}{\partial W_{n'm'}} : W_{n'm'} \in V_{nm} \right\}$$

is uncorrelated, having

$$\mathrm{var}(\Delta W_{nm}) = \sum_{W_{n'm'} \in V_{nm}} \mathrm{var}\left(\frac{\partial L}{\partial W_{n'm'}}\right) = B_{nm}\mathrm{var}\left(\frac{\partial L}{\partial W}\right)$$

The only way to satisfy the initialization conditions formula (8) is to have Bnm = 1. However, this corresponds to circulant matrices of size 1×1, which simply implies that the CirCNN does not compress the original network at all.

In order to achieve a desirable compression rate, it needs to introduce an extra positive scaler α and reformulate the fully connected layer with CirCNN in the reference document [2] in the form of formula (7). It is stressed that the scalar α will be a constant for this layer, and the learnable variables are still v and b. Thus by introducing α, the number of learnable parameters is not increased, and the increment in the number of operations for this layer is negligible.

Assuming that it is required to compress this layer by a factor of B. Then the circulant matrices in W should have size $B \times B$. A calculation as before shows that

$$\mathrm{var}(\Delta W_{nm}) = \alpha^4 B_{nm}\mathrm{var}\left(\frac{\partial L}{\partial W}\right) \tag{9}$$

where Bnm = B. Then to achieve the initialization conditions formula (8), setting

$$\begin{cases} \mathrm{var}(v_{\lambda(n,m)}) = \mathrm{gain} \cdot \sqrt{B} \cdot \frac{2}{M+N}, \\ \alpha = \frac{1}{\sqrt[4]{B}}. \end{cases} \tag{10}$$

where gain should be determined by the activation function. To see the effect of the positive scalar α on the updates of $v_{\lambda(n,m)}$, calculating as follows. Supposing the learning rate of the SGD optimizer is r, then after one forward and backward propagation, having

$$v_{\lambda(n,m)} - \alpha r \sum_{W_{n'm'} \in V_{nm}} \frac{\partial L}{\partial W_{n'm'}} \mapsto v_{\lambda(n,m)} \tag{11}$$

It shows that now the effective learning rate for $v_{\lambda(n,m)}$ changes from r to ar. Thus by formulating each layer as in formula (7), the effective learning rate of learnable parameters layer by layer can be changed.

The effectiveness of the proposed initialization method is shown by numerical experiments.

Example 1. A training process of a simple network with the initialization formula (10) is shown on the MNIST dataset. The network structure is summarized in Table 1, and CirCNN compression is used for the first fully connected layer. It is noted that after the compression, the total number of learnable parameters is only about 0.76% of the original model. From the plot of the loss function in Figure 1, we see that the proposed initialization method better suits this CirCNN network.

Layer	Output channel	Number of parameters	Compression ratio
Conv2d	32	$3 \times 3 \times 1 \times 32$	1
MaxPool			
Conv2d	64	$3 \times 3 \times 32 \times 64$	1
MaxPool			
Fc	1568	3136×1568	1568
Fc	10	1568×10	1

Table 1 Network structure of Example 1. For a compression ratio B>=1, circulant implementation with block size B. The number of parameters for a CirCNN implementation should be divided by B.

Example 2. A simple network where the weight matrix of many layers share a common weight matrix V. Detail of the network, together with the initialization of the common weight V by proposed formula (8), is summarized in formula (12a) and (12b)

$$\begin{cases} y_0 = \mathrm{ReLU}(W_0 x), \\ W_1 = W_2 = \cdots = W_{50}, \\ y_1 = W_1 y_0, \\ \quad \vdots \\ y_{50} = W_{50} y_{49}, \\ y = W y_{50}. \end{cases} \tag{12a}$$

$$\begin{cases} \alpha = 1/\sqrt{50}, \\ \mathrm{var}(V) = 1/(\alpha^2 M). \end{cases} \tag{12b}$$

where as before, for the proposed initialization, the weight matrices have the form $W_i = \alpha V$ for i = 1, 2, …, 50. A comparison with the initialization (12b) and the Xavier/He initialization is summarized in Figure 2.

Example 3. The proposed initialization method on a CirCNN implementation of the VGG16 network is tested. For the second fully connected layer in the VGG16 network which has 4096 input channels and 4096 output channels, a circulant matrix with block size 4096 is used. The Cifar10 data set with 5 runs for both Xavier/He and the proposed initialization methods are tested. The original VGG16 network can achieve 92.24% top-1 accuracy. On the 5 runs, the network with proposed initialization consistently outperforms the network with the Xavier/He initialization on top-1 accuracy. The average top-1 accuracy for proposed method is 92.00%, while the average top-1 accuracy for the Xavier/He initialization is 90.98%.

Figure 3 illustrates a method for a neutral network training. The training solution with the proposed initialization method can be summarized as following:

S101: Determining parameters for an initialization weight matrix of a layer of a neural network, according to the block size in circulant implementation of the layer.

It is noted that for another structured implementation with shared elements, S101 comprises determining parameters for a weight matrix of each layer of a network, based on the number of shared elements in the structured implementation with shared elements of each layer.

In an implementation, the parameters include:

$$\begin{cases} \mathrm{Var}\left(v_{\lambda(n,m)}\right) = \mathrm{gain} \cdot \sqrt{B_{nm}} \cdot \frac{2}{M+N} \\ C_{nm} = \frac{1}{\sqrt[4]{B_{nm}}} \end{cases}$$

$$\mathrm{Std}\left(v_{\lambda(n,m)}\right) = \sqrt{\mathrm{Var}\left(v_{\lambda(n,m)}\right)} \cdot$$

where gain equals to 1 for Identity Function, and equals to 2 in the initialization of ReLU, M or m represents the number of input channels, N or n represents the number of output channels, Bnm represents the number of parameter sharing of Wnm, which depends on the method of parameter sharing. For example, a circulant matrix mentioned above is a matrix with parameter sharing form. In a circnn network, ∀n,m, Bnm=B, wherein B is the block size in circulant implementation of the layer. Xavier_value equals to $\mathrm{Std}\left(v_{\lambda(n,m)}\right) \cdot \sqrt{3.0}$, which is used to generate $v_{\lambda(n,m)}$, a random number in an uniform distribution between -Xavier_value and Xavier_value, that is [-Xavier_value, Xavier_value].

It is noted that Cnm is negatively correlated with the number of shared weights. The more weights sharing, the smaller value of Cnm. In this case, the learning rates of the shared weights become lower, the stability of training the neural network is improved.

And in some other embodiments, Cnm can be equal to 1/Bnm, 1/(Bnm × Bnm) and so on.

S102: Determining the initialization weight matrix according to the parameters.

More specifically,

$$W_{nm} = C_{nm} \cdot v_{\lambda(n,m)}$$

S103: Decompressing Wnm to derive the integrated block-circulant matrix Wt.

Figure 4 exemplary shows the compression of a block-circulant matrix. The decompression is an inverse process or the compression. And in a lossless compression, it is an absolute inverse process.

S104: Forward propagation of training.

In an implementation, the training date are trained by formula (1), which is Y=Wt · X+B, where X represents the input date, Y represents the output date, B represents the bias, and Wt is derived from step S103. And it is noted that when the layer is a convolution layer, X and B are date translated by "im2col" operation.

S105: Backward propagation of training.

In an implementation, the output date of step S104 are trained by

$$\frac{\partial \mathrm{L}}{\partial \mathrm{X}} = \sum \mathrm{W'_t} \cdot \frac{\partial \mathrm{L}}{\partial \mathrm{Y}}$$

where L is the loss function of X and Y, $\mathrm{W'_t}$ is an updated value of Wt. Accordingly, $v_{\lambda(n,m)}$is updated.

S106: Performing iteration of step S104 and S105 until the result of training is convergence.

It is noted that the criterion of the convergence is not limited in the present application. For example, the criterion can be the difference of any parameter ($v_{\lambda(n,m)}$, Wt, loss and so on) involved in the training between two neighboring iteration is smaller than a threshold.

S107: Storing the trained parameters of the neural network.

In an implementation, steps S101-S103 are implemented on each layer of the neural network, and steps S104-S107 are implemented based on the whole network.

It is noted that the training date can be images, videos, voice signal, shape, color and other to-be-recognized information, and accordingly, the trained neural network can be used for the related applications, for example, object classifications, natural language processing, and speech recognition, which are mentioned in the above descriptions.

In an embodiment of the present application, a neutral network training method is disclosed, comprising: determining parameters for a weight matrix of each layer of a network, based on the number of shared elements in a structured implementation with shared elements of each layer; determining the weight matrix of each layer based on the determined parameter; decompressing the determined weight matrix to derived an integrated matrix; iterating over forward propagation and backward propagation on all layers of the network based on the integrated matrix until a preset condition is satisfied, wherein the integrated matrix is updated in each iteration; and storing the iterated parameters of the network.

In a feasible implementation, the parameters comprises an adjusting parameter, and wherein the adjusted parameter is negatively correlated with the number of shared elements.

In a feasible implementation, the number of shared elements in the structured implementation with shared elements is a block size in a circulant implementation.

In a feasible implementation, the adjusting parameter is calculated as following: $C_{nm} = \frac{1}{\sqrt[4]{B_{nm}}}$, and wherein m is a number of input channel of a layer of the network, n is a number of output channel of the layer of the network, Cnm is the adjusting parameter, and Bnm is the number of shared elements.

In a feasible implementation, the parameters further comprises a random number in an uniform distribution between -Xavier_value and Xavier_value.

In a feasible implementation, Xavier_value is determined by the following:

$v_{\lambda(n,m)} = \mathrm{Std}\left(v_{\lambda(n,m)}\right) * \sqrt{3.0}$

where $\mathrm{Std}\left(v_{\lambda(n,m)}\right) = \sqrt{\mathrm{Var}\left(v_{\lambda(n,m)}\right)}$, $\mathrm{Var}\left(v_{\lambda(n,m)}\right) = \mathrm{gain} \cdot \sqrt{B_{nm}} \cdot \frac{2}{m+n}$, gain is a consist number, $v_{\lambda(n,m)}$is the random number.

In a feasible implementation, the weight matrix is determined by the following:

$W_{nm} = C_{nm} \cdot v_{\lambda(n,m)}$

where W_{nm} is the weight matrix.

In a feasible implementation, the integrated matrix is updated in each iteration, comprises the random number is updated in each iteration.

In a feasible implementation, the preset condition comprises a training result of the network is convergence.

In a feasible implementation, the training result of the network is convergence, comprises a difference of the weight matrix is smaller than a preset threshold.

In a feasible implementation, an input of the network in training comprises image information data or sound information data.

In a feasible implementation, the network is used to classify object, process language or recognize speech.

In an embodiment of the present application, as shown in Figure 5, a neutral network training device 400 is disclosed, comprising: a first calculation module 401, to determine parameters for a weight matrix of each layer of a network, based on the number of shared elements in a structured implementation with shared elements of each layer; a second calculation module 402, to determine the weight matrix of each layer based on the determined parameter; a decompression module 403, to decompress the determined weight matrix to derived an integrated matrix; an iteration module 404, to iterate over forward propagation and backward propagation on all layers of the network based on the integrated matrix until a preset condition is satisfied, wherein the integrated matrix is updated in each iteration; and a storage module 405, to store the iterated parameters of the network.

In a feasible implementation, the parameters comprises an adjusting parameter, and wherein the adjusted parameter is negatively correlated with the number of shared elements.

In a feasible implementation, the number of shared elements in the structured implementation with shared elements is a block size in a circulant implementation.

In a feasible implementation, the adjusting parameter is calculated as following: $C_{nm} = \frac{1}{\sqrt[4]{B_{nm}}}$, and wherein m is a number of input channel of a layer of the network, n is a number of output channel of the layer of the network, Cnm is the adjusting parameter, and Bnm is the number of shared elements.

In a feasible implementation, the parameters further comprises a random number in an uniform distribution between -Xavier_value and Xavier_value.

In a feasible implementation, Xavier_value is determined by the following:

$v_{\lambda(n,m)} = \mathrm{Std}\left(v_{\lambda(n,m)}\right) * \sqrt{3.0}$

where $\mathrm{Std}\left(v_{\lambda(n,m)}\right) = \sqrt{\mathrm{Var}\left(v_{\lambda(n,m)}\right)}$, $\mathrm{Var}\left(v_{\lambda(n,m)}\right) = \mathrm{gain} \cdot \sqrt{B_{nm}} \cdot \frac{2}{m+n}$, gain is a consist number, $v_{\lambda(n,m)}$is the random number.

In a feasible implementation, the weight matrix is determined by the following:

$W_{nm} = C_{nm} \cdot v_{\lambda(n,m)}$

where W_{nm} is the weight matrix.

In a feasible implementation, the integrated matrix is updated in each iteration, comprises the random number is updated in each iteration.

In a feasible implementation, the preset condition comprises a training result of the network is convergence.

In a feasible implementation, the training result of the network is convergence, comprises a difference of the weight matrix is smaller than a preset threshold.

In a feasible implementation, an input of the network in training comprises image information data or sound information data.

In a feasible implementation, the network is used to classify object, process language or recognize speech.

In an embodiment of the present application, an apparatus for training a neutral network is disclosed, the apparatus comprising: one or more processors; and a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by the processors, configures the apparatus to carry out the method according to any one of claims 1-12 in the present application.

In an embodiment of the present application, a computer program product is disclosed, comprising a program code for performing the method according to any one of claims 1-12 in the present application.

Figure 6 is a simplified block diagram of an apparatus **500** that may be used as the apparatus for training a neutral network according to the exemplary embodiment.

A processor 502 in the apparatus 500 can be a central processing unit. Alternatively, the processor 502 can be any other type of device, or multiple devices, capable of manipulating or processing information now-existing or hereafter developed. Although the disclosed implementations can be practiced with a single processor as shown, e.g., the processor 502, advantages in speed and efficiency can be achieved using more than one processor.

A memory 504 in the apparatus 500 can be a read only memory (ROM) device or a random access memory (RAM) device in an implementation. Any other suitable type of storage device can be used as the memory 504. The memory 504 can include code and data 506 that is accessed by the processor 502 using a bus 512. The memory 504 can further include an operating system 508 and application programs 510, the application programs 510 including at least one program that permits the processor 502 to perform the methods described here. For example, the application programs 510 can include applications 1 through N, which further include a video coding application that performs the methods described here.

The apparatus 500 can also include one or more output devices, such as a display 518. The display 518 may be, in one example, a touch sensitive display that combines a display with a touch sensitive element that is operable to sense touch inputs. The display 518 can be coupled to the processor 502 via the bus 512.

Although depicted here as a single bus, the bus 512 of the apparatus 500 can be composed of multiple buses. Further, the secondary storage 514 can be directly coupled to the other components of the apparatus 500 or can be accessed via a network and can comprise a single integrated unit such as a memory card or multiple units such as multiple memory cards. The apparatus 500 can thus be implemented in a wide variety of configurations.

In some embodiments, some or all of the functions or processes of the one or more of the devices are implemented or supported by a computer program that is formed from computer readable program code and that is embodied in a computer readable medium. The phrase "computer readable program code" includes any type of computer code, including source code, object code, and executable code. The phrase "computer readable medium" includes any type of

medium capable of being accessed by a computer, such as read only memory (ROM), random access memory (RAM), a hard disk drive, a compact disc (CD), a digital video disc (DVD), or any other type of memory.

It may be advantageous to set forth definitions of certain words and phrases used throughout this patent document. The terms "include" and "comprise," as well as derivatives thereof, mean inclusion without limitation. The term "or" is inclusive, meaning and/or. The phrases "associated with" and "associated therewith," as well as derivatives thereof, mean to include, be included within, interconnect with, contain, be contained within, connect to or with, couple to or with, be communicable with, cooperate with, interleave, juxtapose, be proximate to, be bound to or with, have, have a property of, or the like.

While this disclosure has described certain embodiments and generally associated methods, alterations and permutations of these embodiments and methods will be apparent to those skilled in the art. Accordingly, the above description of example embodiments does not define or constrain this disclosure. Other changes, substitutions, and alterations are also possible without departing from the scope of this disclosure, as defined by the following claims.

CLAIMS

1. A neutral network training method, comprising:

determining parameters for a weight matrix of each layer of a network, based on the number of shared elements in a structured implementation with shared elements of each layer;

determining the weight matrix of each layer based on the determined parameter;

decompressing the determined weight matrix to derived an integrated matrix;

iterating over forward propagation and backward propagation on all layers of the network based on the integrated matrix until a preset condition is satisfied, wherein the integrated matrix is updated in each iteration; and

storing the iterated parameters of the network.

2. The method of claim 1, wherein the parameters comprises an adjusting parameter, and wherein the adjusted parameter is negatively correlated with the number of shared elements.

3. The method of claim 1 or 2, wherein the number of shared elements in the structured implementation with shared elements is a block size in a circulant implementation.

4. The method of any one of claims 1-3, wherein the adjusting parameter is calculated as following: $C_{nm} = \frac{1}{\sqrt[4]{B_{nm}}}$, and wherein m is a number of input channel of a layer of the network, n is a number of output channel of the layer of the network, Cnm is the adjusting parameter, and Bnm is the number of shared elements.

5. The method of claim 4, wherein the parameters further comprises a random number in an uniform distribution between -Xavier_value and Xavier_value.

6. The method of claim 5, wherein Xavier_value is determined by the following:

$v_{\lambda(n,m)} = \mathrm{Std}\left(v_{\lambda(n,m)}\right) * \sqrt{3.0}$

where $\mathrm{Std}\left(\mathrm{v}_{\lambda(\mathrm{n,m})}\right) = \sqrt{\mathrm{Var}\left(\mathrm{v}_{\lambda(\mathrm{n,m})}\right)}$, $\mathrm{Var}\left(\mathrm{v}_{\lambda(\mathrm{n,m})}\right) = \mathrm{gain} \cdot \sqrt{\mathrm{B}_{\mathrm{nm}}} \cdot \frac{2}{\mathrm{m+n}}$, gain is a consist number, $v_{\lambda(n,m)}$is the random number.

7. The method of claim 6, wherein the weight matrix is determined by the following:

$W_{nm} = C_{nm} \cdot v_{\lambda(n,m)}$

where W_{nm} is the weight matrix.

8. The method of any one of claims 5-7, wherein the integrated matrix is updated in each iteration, comprises the random number is updated in each iteration.

9. The method of any one of claims 5-8, wherein the preset condition comprises a training result of the network is convergence.

10. The method of claim 9, wherein the training result of the network is convergence, comprises a difference of the weight matrix is smaller than a preset threshold.

11. The method of any one of claims 1-10, wherein an input of the network in training comprises image information data or sound information data.

12. The method of any one of claims 1-11, wherein the network is used to classify object, process language or recognize speech.

13. A neutral network training device, comprising:

a first calculation module, to determine parameters for a weight matrix of each layer of a network, based on the number of shared elements in a structured implementation with shared elements of each layer;

a second calculation module, to determine the weight matrix of each layer based on the determined parameter;

a decompression module, to decompress the determined weight matrix to derived an integrated matrix;

an iteration module, to iterate over forward propagation and backward propagation on all layers of the network based on the integrated matrix until a preset condition is satisfied, wherein the integrated matrix is updated in each iteration; and

a storage module, to store the iterated parameters of the network.

14. The device of claim 13, wherein the parameters comprises an adjusting parameter, and wherein the adjusted parameter is negatively correlated with the number of shared elements.

15. The device of claim 13 or 14, wherein the number of shared elements in the structured implementation with shared elements is a block size in a circulant implementation.

16. The device of any one of claims 13-15, wherein the adjusting parameter is calculated as following: $C_{nm} = \frac{1}{\sqrt[4]{B_{nm}}}$, and wherein m is a number of input channel of a layer of the network, n is a number of output channel of the layer of the network, Cnm is the adjusting parameter, and Bnm is the number of shared elements.

17. The device of claim 16, wherein the parameters further comprises a random number in an uniform distribution between -Xavier_value and Xavier_value.

18. The device of claim 17, wherein Xavier_value is determined by the following:

$v_{\lambda(n,m)} = \mathrm{Std}\left(v_{\lambda(n,m)}\right) * \sqrt{3.0}$

where $\mathrm{Std}\left(v_{\lambda(n,m)}\right) = \sqrt{\mathrm{Var}\left(v_{\lambda(n,m)}\right)}$, $\mathrm{Var}\left(v_{\lambda(n,m)}\right) = \mathrm{gain} \cdot \sqrt{B_{nm}} \cdot \frac{2}{m+n}$, gain is a consist number, $v_{\lambda(n,m)}$is the random number.

19. The device of claim 18, wherein the weight matrix is determined by the following:

$W_{nm} = C_{nm} \cdot v_{\lambda(n,m)}$

where W_{nm} is the weight matrix.

20. The device of any one of claims 17-19, wherein the integrated matrix is updated in each iteration, comprises the random number is updated in each iteration.

21. The device of any one of claims 17-20, wherein the preset condition comprises a training result of the network is convergence.

22. The device of claim 21, wherein the training result of the network is convergence, comprises a difference of the weight matrix is smaller than a preset threshold.

23. The device of any one of claims 13-22, wherein an input of the network in training comprises image information data or sound information data.

24. The device of any one of claims 13-23, wherein the network is used to classify object, process language or recognize speech.

25. An apparatus for training a neutral network, the apparatus comprising:

one or more processors; and

a non-transitory computer-readable storage medium coupled to the processors and storing programming for execution by the processors, wherein the programming, when executed by the processors, configures the apparatus to carry out the method according to any one of claims 1-12.

26. A computer program product comprising a program code for performing the method according to any one of claims 1-12.

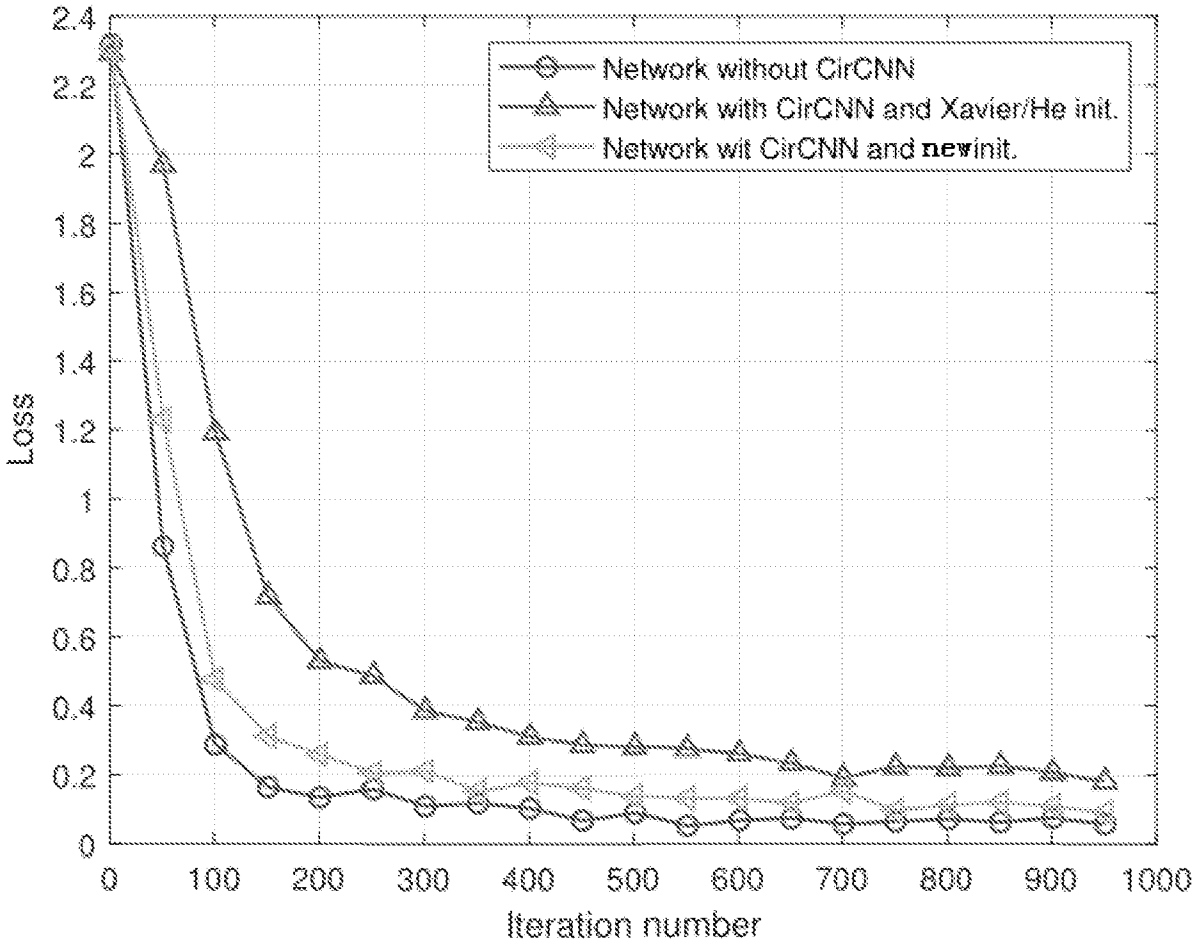
Network without CirCNN
Network with CirCNN and Xavier/He init.
Network wit CirCNN and newinit.
Loss
Iteration number
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
0
100
200
300
400
500
600
700
800
900
1000

Figure 1

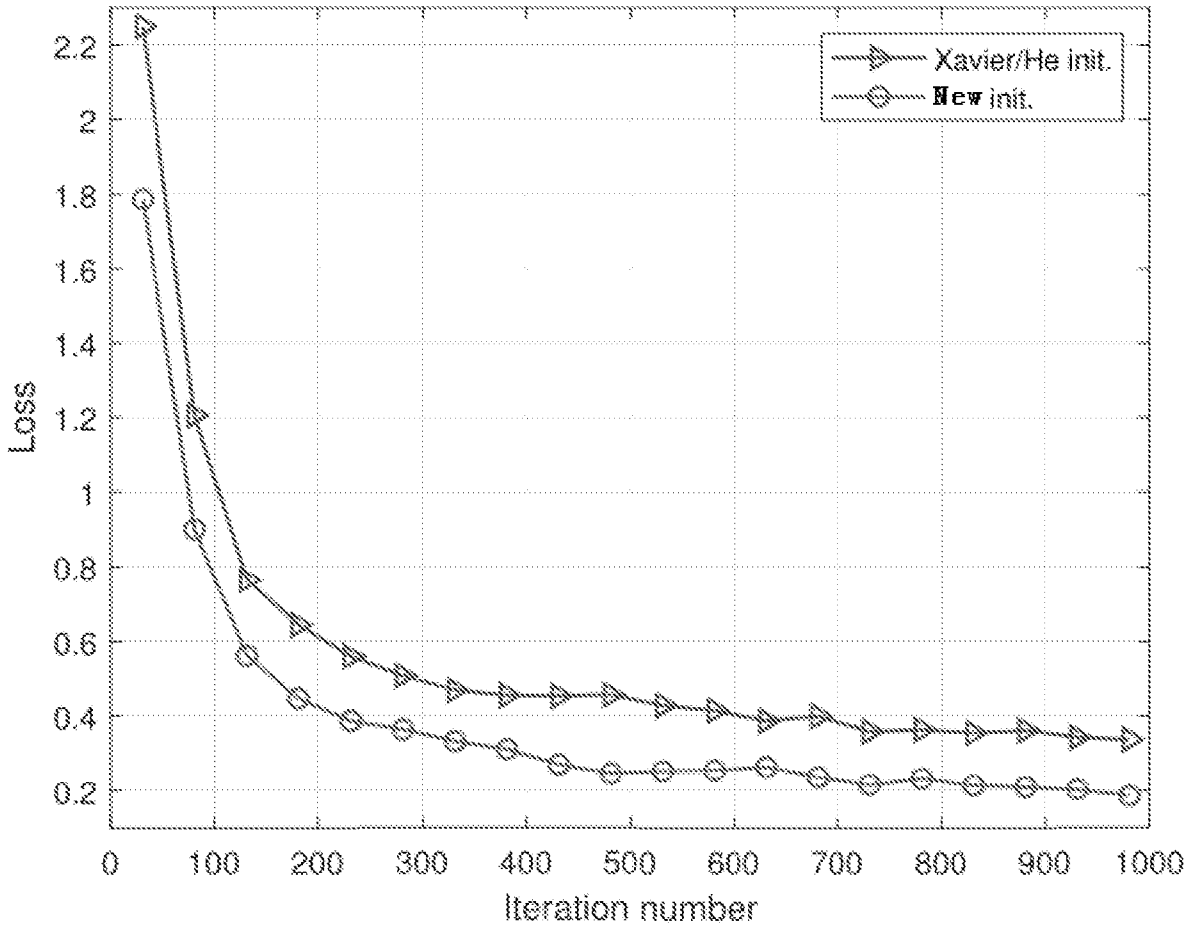
Xavier/He init.
New init.
Loss
Iteration number
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
100
200
300
400
500
600
700
800
900
1000

Figure 2

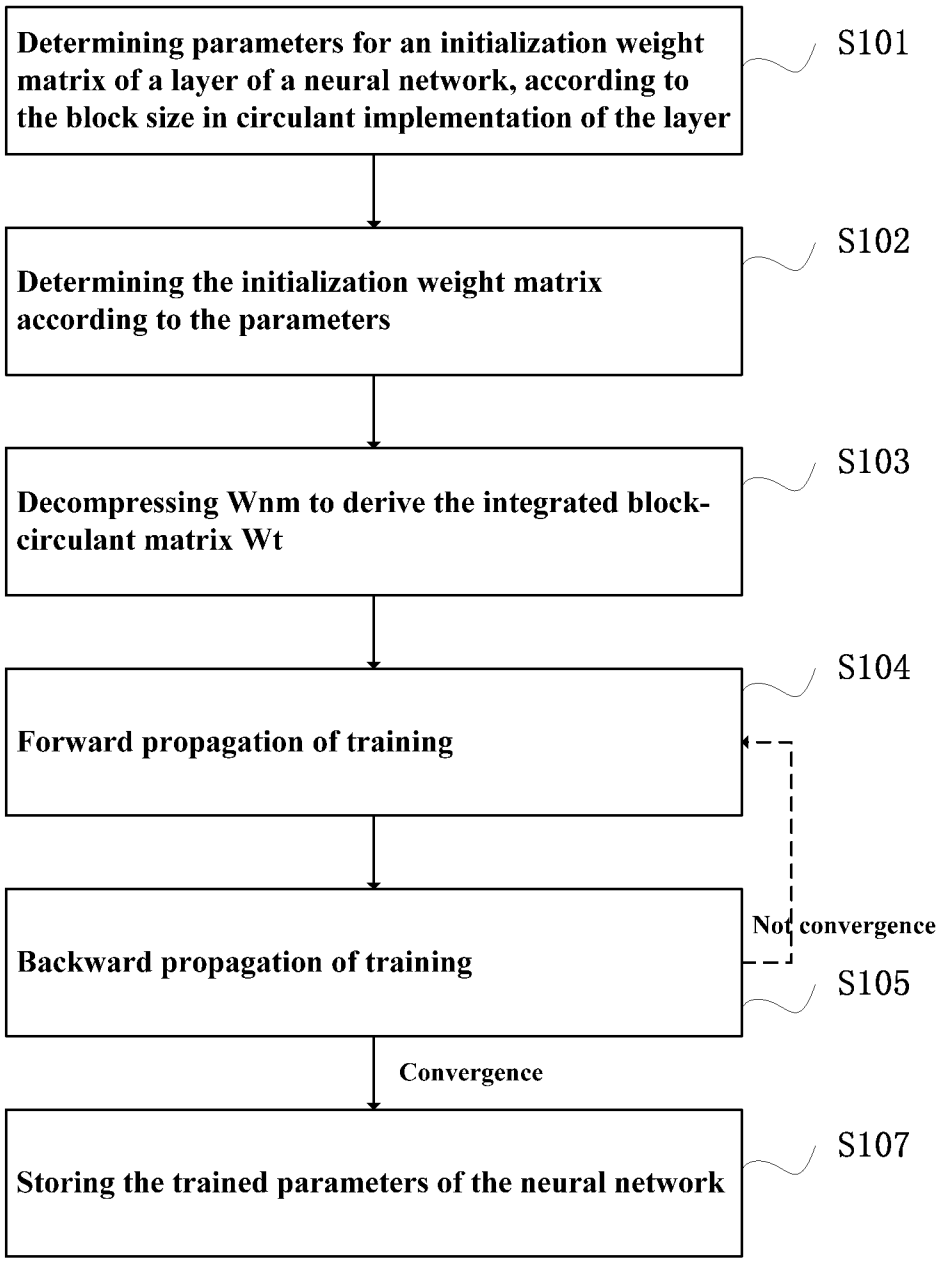
Determining parameters for an initialization weight matrix of a layer of a neural network, according to the block size in circulant implementation of the layer
S101
Determining the initialization weight matrix according to the parameters
S102
Decompressing Wnm to derive the integrated block-circulant matrix Wt
S103
Forward propagation of training
S104
Backward propagation of training
Not convergence
S105
Convergence
Storing the trained parameters of the neural network
S107

Figure 3

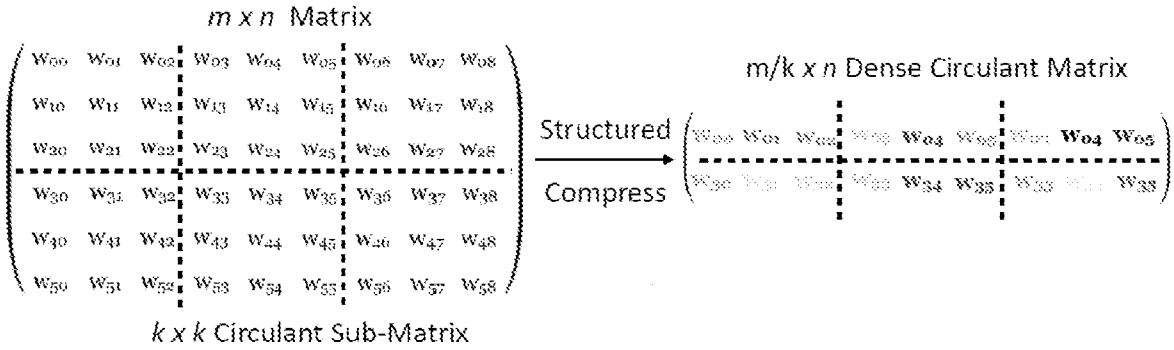

Figure 4

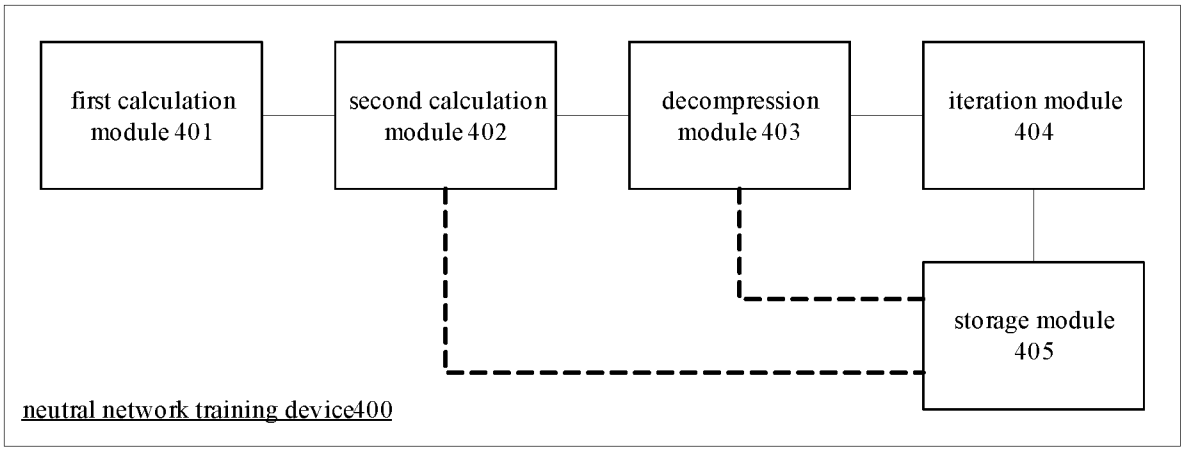

Figure 5

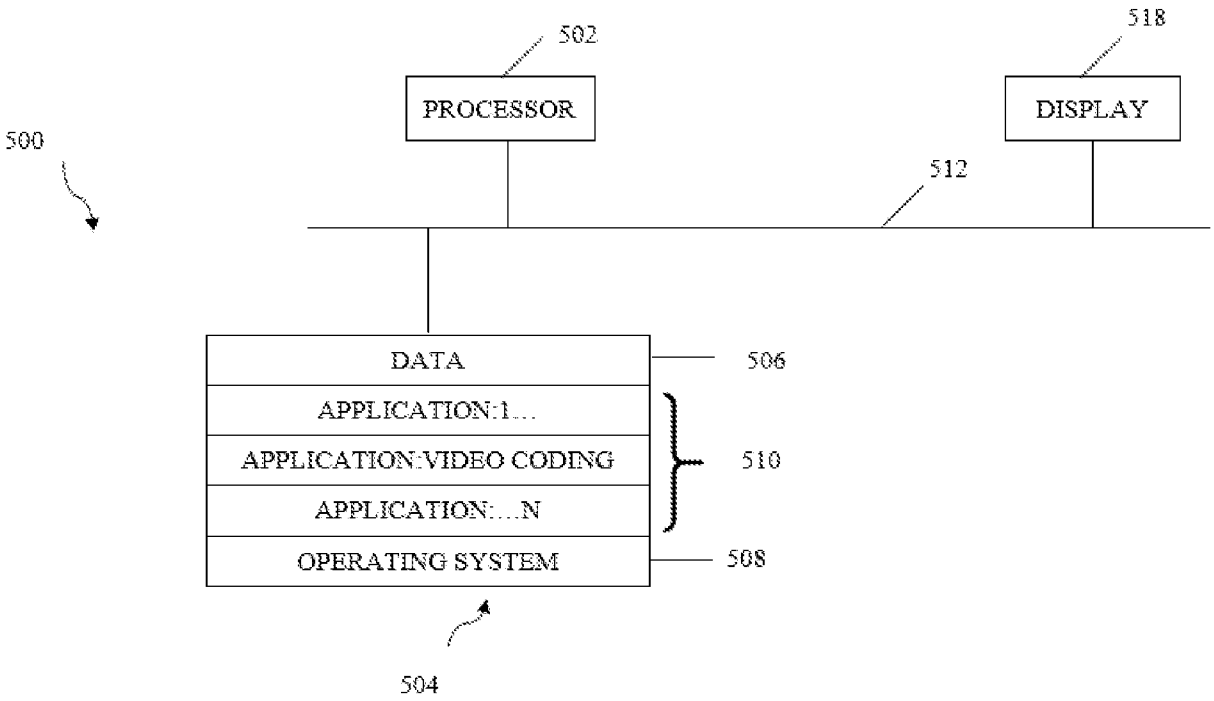
502
518
PROCESSOR
DISPLAY
500
512
DATA
506
APPLICATION:1...
APPLICATION:VIDEO CODING
510
APPLICATION:...N
OPERATING SYSTEM
508
504

Figure 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2019/084969

A. CLASSIFICATION OF SUBJECT MATTER

G06N 3/08(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06N; G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

DWPI;CNABS;CNTXT;USTXT;CNKI;SIPOABS:nuetral network, train, weight matrix, layer, integrated matrix, forward, backward, propagation

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 104899641 A (HANGZHOU LANGHE TECHNOLOGY CO., LTD.) 09 September 2015 (2015-09-09) the whole document	1-26
A	US 2017103308 A1 (IBMC INT BUSINESS MACHINES CORP) 03 April 2017 (2017-04-03) the whole documnet	1-26

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
23 January 2020	**12 February 2020**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer **CUI,Liyan** Telephone No. **(86-10) 62411682**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2019/084969

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 104899641 A	09 September 2015	CN 104899641 B	13 July 2018
US 2017103308 A1	03 April 2017	US 10380479 B2	13 August 2019