PCA Based Kernel Initialization for Convolutional Neural Networks

Yifeng Wang[1], Yuxi Rong[1], Hongyue Pan[1], Ke Liu[1], Yang Hu[2], Fangmin Wu[2], Wei Peng[2], Xingsi Xue[3], and Junfeng Chen[4(✉)]

[1] China Three Gorges Corporation, Beijing 100038, China
{wang_yifeng1,rong_yuxi,pan_hongyue,liu_ke1}@ctg.com.cn
[2] Hangzhou HuaNeng Engineering Safety Technology Co., Ltd., Hangzhou 311121, China
mark_hnsafet@qq.com, 329537060@qq.com, 12492687@qq.com
[3] Fujian Key Lab for Automotive Electronics and Electric Drive, Fujian University of Technology, Fuzhou 350118, Fujian, China
jack8375@gmail.com
[4] College of IoT Engineering, Hohai University, Changzhou 213022, Jiangsu, China
chen-1997@163.com

Abstract. The initialization of Convolutional Neural Networks (CNNs) is about providing reasonable initial values for the convolution kernels and the fully connected layers. In this paper, we proposed a convolution kernel initialization method based on the two-dimensional principal component analysis (2DPCA), in which a parametric equalization normalization method is used to adjust the scale between each neuron weight. After that the weight initial value can be adaptively adjusted according to different data samples. This method enables each neuron to fully back-propagate errors and accelerate network model training. Finally, a network model was built and experiments were performed using $Tanh$ and $ReLU$ activation functions. The experimental results verify the effectiveness of the proposed method through the distribution of histograms and the curve comparison diagrams of model training.

Keywords: Convolutional neural networks · Convolution kernel initialization · PCA · Parametric equalization normalization

1 Introduction

The Convolutional Neural Networks (CNNs) [2], as representative deep learning models, have a remarkable ability to extract features directly from the original images and recognize the rules of these visual images with minimal preprocessing. The most common form of the CNNs architecture stacks a number of convolutional and pooling layers optionally followed by fully connected layers. Most notable among these is the convolution kernel which is a small trainable matrix used for features detection. The initialization of CNNs is about providing reasonable initial values for the convolution kernels and the fully connected layers.

Y. Tan et al. (Eds.): DMBD 2020, CCIS 1234, pp. 71–82, 2020.
https://doi.org/10.1007/978-981-15-7205-0_7

The kernel initialization of the CNNs is an issue worthy of discussion. For simple Neural Networks (NN), random initialization would be a good choice. G. Thimm and E. Fiesler designed experiments to test the random weight initialization methods on the multilayer perceptron and the high order networks [8]. He et al. found that the rectifying activation unit is very important for the neural network, and proposed a Parametric Rectified Linear Unit (PReLU) to generalize the traditional rectified units. It is well to be reminded that they derived a robust initialization method, particularly considering the rectifier nonlinearities [1]. Sun et al. proposed Multi-layer Maxout Networks (MMN) with multi-layer which can train active function, and deduced a new initialization method dedicated to the activation of MMN. The method can reduce the movement of internal covariates when the signal propagates through the layer [6]. A. Pacheco determined the input weights and bias for the Extreme Learning Machine (ELM) by using the Restricted Boltzmann Machine (RBM), named as RBM-ELM [4]. Similarly, Zhang and Ji also constructed a RBM model to pre-train convolution kernels. The trained weight matrix is transformed to initialize the convolution kernel parameters of the CNNs [11]. Liu et al. proposed an image extraction algorithm by mixing the AutoEncoder and the CNNs. They utilized the AutoEncoder to train the basic elements of image and initialized the convolution kernel of the CNNs [3]. Yang et al. used sparse coding to extract the convolution kernel for initialization, which can shorten the training time and raise the recognition rate [9]. In target super resolution, Li et al. proposed a Multi-channel Convolution image Super-Resolution (MCSR) algorithm, which used a residual CNN based on sparse coding and an MSRA initialization method to accelerate model training the convergence [10]. M. S. Seyfioğlu compared two NN initialization methods, unsupervised pre-training and transfer-learning, in training the deep NN on small data sets [5]. Tang et al. employed k-means unsupervised feature learning as the pre-training process, instead of the traditional random initialization weights [7]. So far, there are no unified understanding and development methods for the initialization problem of the CNNs. Moreover, the current initialization methods gave no considerations to the sample information and cannot automatically adapt to variation of the samples. In this paper, we attempt to employ the Principal Component Analysis (PCA) into kernel initialization and propose a parametric equalization normalization to adjust the scale among the neuron weights.

The rest of the article is organized as follows. Section 2 reviews the methods of the convolution kernel initialization. Section 3 proposed the PCA-based convolution kernel initialization with balanced normalization. Section 4 presents the experimental configuration and results; finally, the conclusions are presented in Sect. 5.

2 Convolution Kernel Initialization

There are mainly three initialization methods: random initialization, Xavier initialization and MSRA Initialization.

2.1 Random Initialization

The random initialization method generally refers to the normal distribution randomization method. It defines a random variable x, which obeys a probability distribution $f(x)$ with mean μ and standard deviation σ. We have the following probability density function.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{1}$$

The random variable distribution obeys the normal distribution, referred to as $f(x) \sim N(\mu, \sigma^2)$ are set as different values. The mean μ affects the symmetry center of the curve, and the standard deviation σ influences the smoothness of the curve. The larger the μ is, the smoother the curve will be. The random initialization method initializes each parameter of the convolution kernel according to the Gaussian probability distribution.

2.2 Xavier Initialization

Xavier Glorot et al. [12] proposed the Xavier initialization method whose core idea is to keep the variance of input and output consistent and prevent all output values from going to 0. The literature derivation gives the specific form of Xavier initialization:

$$W \sim U\left(-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right) \tag{2}$$

where n_j is the total number of input neurons for each feature map of the layer, and n_{j+1} is the total number of output neurons. The U denotes a uniform distribution within this range. Each weight parameter is randomly initialized to random value which obeys this uniformly distribution. The parameters of the method only depend on the number of input and output neurons, do not need to manually set parameters, and are completely adaptive to the size of the network model itself, and are more stable and easy to use than the random initialization method.

2.3 MSRA Initialization

The MSRA initialization method is an improved method for Xavier. It is particularly applicable to the most popular *Relu* function. Its specific form is shown in (3). It is a normal distribution with a mean 0 and a variance $2/n$.

$$W \sim G\left[0, \sqrt{\frac{2}{n}}\right] \tag{3}$$

where n is the number of inputs to the layer, and G denotes that obeys a normal distribution. The parameters of this method are only dependent on the number of neurons of one input, less than the two parameters of Xavier, and are particularly suitable for *Relu* functions.

3 PCA-Based Convolution Kernel Initialization with Balanced Normalization

In this section, we employ the Two-Dimensional Principal Component Analysis (2DPCA) to extract the feature vectors of the image and introduce an equalization normalization method to adjust the scale of the weight between layers.

3.1 Two-Dimensional Principal Component Analysis

Suppose there is an image sample set $X = \{X_1, X_2, \cdots, X_N\}$, the number of sample sets is N, and the dimension is $m \times n$. Projecting images in both row and column directions, so that the variance remaining in the subspace is the largest. The fact that 2DPCA uses orthogonal transformation to eliminate the correlation between the original vectors should be paid attention too. The two principal components obtained are linearly independent. Therefore, the principal component may represent higher information than the original data. The specific 2DPCA calculation steps are as follows:

(1) Calculate the average of all image samples: $\overline{X}=\frac{1}{N}\sum_{i=1}^{N} X_i$;
(2) Calculate the covariance matrix: $G_t=\frac{1}{N}\sum\limits_{j=1} (X_i - \overline{X})^T (X_i - \overline{X})$;
(3) The eigenvalue decomposition of the covariance matrix G_t is performed by using the Jacobian method to obtain the corresponding eigenvalue $\lambda_1, \lambda_2, \cdots, \lambda_d$ and the eigenvector corresponding to each eigenvalue. The eigenvector corresponding to the largest k eigenvalues is selected to compose the projection matrix $U=[u_1, u_2, \cdots, u_k] \in R^{n\times k}$;
(4) Do feature extraction to every sample data by column: $F_j = X_j \cdot U \in R^{m\times k}$, what is obtained is the feature of X_j. The original two-dimensional $m \times n$ size image is now reduced to $m \times k$, that is, the number of bits of the matrix column vectoris compressed after the feature extraction, and the number of row is not changed;
(5) After the above process, a new sample F_j, $j = 1, 2, \cdots, N$ is obtained. On the new sample, repeat step (1) (2) construct a new covariance matrix G_t^*: $G_t^* = \frac{1}{N}\sum\limits_{i=1}^{N} \left(F_i - \overline{F}\right)^T \left(F_i - \overline{F}\right)$;
(6) Similarly, repeat step (3) to take the feature vector corresponding to the largest d eigenvalues to obtain the projection matrix in the row direction $V=[v_1, v_2, \cdots, v_d]$.

Since the variance of each principal component is gradually reduced, the key information included is also decremented. So generally, the contribution rate can indicate the amount of information occupied by a principal component. Specifically, it refers to the proportion of the cumulative value of the total variance of a principal component, also, the proportion of the sum of a feature value to the sum of all feature values, which is:

$$\eta = \frac{\lambda_i}{\sum\limits_{i=1}^{d} \lambda_i} \tag{4}$$

3.2 2DPCA Initialization

The 2DPCA method is used to initialize each convolution kernel in the network. The initialization is based on training sample data. The size of the convolution kernel is $R_k \times R_k$. The size of the input map for each layer is set to $C_k \times C_k$. There are a total of D_k maps. The process is as follows.

(1) Manually select one picture for each category from all the data. That is, a total of n pictures can be selected. Each picture can correspond to a category of its own.
(2) Count the number of neurons input and output for each convolutional layer, and use the Xavier initialization method to initialize all the parameter weights $W_{k,i,j}$ for each convolutional layer in turn. The $W_{k,i,j}$ denotes the weight of the j-th window of the i-th convolution kernel in the k-th layer, and all offsets are initialized to 0.
(3) Do forward-propagating to each picture, and obtain the corresponding feature map $C_{k,i,z}$, $z \in [1, N]$, which denotes the i-th feature map of the k-th layer of the z-th picture. The size r_k of the feature map obtained between different layers is different. The inputting size of picture of each layer is $C_k - R_k + 1$.
(4) Combine the feature maps of the same location corresponding to different pictures into a set $P_{k,i}$, which indicates the i-th feature map of the k-th layer, and sample the $R_k \times R_k$ window size of all the graphs in the set to obtain the image set $P_{k,i}$. And the number of sampling result in $P_{k,i}$ is $n \times (r_k - R_k + 1) \times (r_k - R_k + 1)$.
(5) Calculating all the images in the set $P_{k,j}$ according to the 2DPCA steps in 3.3.1, then obtain the corresponding projection matrix $U_{k,j}, V_{k,j} \in R^{R_k \times R_k}$, which correspond to the two columns and rows, and the feature value sets λ_u and λ_v corresponding to each feature vector.
(6) After arbitrarily adding the eigenvalues in λ_u and λ_v get a new set λ^*, take the largest D_k values before. Every value corresponds to the two eigenvectors ξ_u^a and ξ_v^b respectively. Calculating $\xi_v^b * \xi_u^{aT}$ in turn yields a set of evaluation matrices $M = \{M_1, M_2, \cdots M_T\}$, where the maximum value of T is R_k^2. Then the initialize the convolution kernel in turn.

3.3 Parametric Equalization Normalization

For an N-layer convolutional neural network, the loss function is defined as $\ell(z_N)$. We usually want the gradient of the weight $W_k(i,j)$ of the k-th layer to satisfy the following form:

$$C_{k,i,j}^2 = E_{z_0} \sim D\left[\frac{\partial}{\partial W_k(i,j)}\ell(z_N)\right] = E_{z_0} \sim D\left[z_{k-1}(j)\frac{\partial}{\partial z_k(i)}\ell(z_N)\right] \quad (5)$$

where D is the set of input images and $y_k = \frac{\partial}{\partial z_k}\ell(z_N)$ is the backpropagation error. A similar formula can also be applied to offset b_k, except that the coefficient becomes constant 1.

In order to make all parameters be able to learn at the same speed, we need to scale (5) proportionally, hoping to be a constant for all weights:

$$\overline{C}_{k,i,j}^2 = \frac{C_{k,i,j}^2}{\|W_k\|_2^2} \tag{6}$$

Among them, $\|W_k\|_2^2$ represents the 2-norm square of matrix W_k, but due to the effect of nonlinear activation function, this condition is difficult to control and guarantee, and the change of weight will directly affect the final output value y_k. We therefore need to simplify (6) so that each convolution kernel in the same layer W_k satisfies about a constant, rather than strictly for all weights:

$$\overline{C}_{k,j}^2 = \frac{1}{N}\sum_i \overline{C}_{k,i,j}^2 = \frac{1}{N\|W_k\|_2^2} E_{z_0} \sim D\left[z_{k-1}^2(j)\|y_k\|_2^2\right] \tag{7}$$

where N is the number of rows of the weight matrix. This formula makes all the values in the same weight matrix have the same trend. At the same time, we can note that for the input in the layer has little effect on the gradient in the entire network. It can be seen that $z_{k-1}(j)$ and $\|y_k\|$ are independent of each other, so we can further simplify the objective function:

$$\overline{C}_{k,j}^2 \approx E_{z_0} \sim D\left[z_{k-1}(j)^2\right] \frac{E_{z_0} \sim D\left[\|y_k\|_2^2\right]}{N\|W_k\|_2^2} \tag{8}$$

The method taking the approximate value is convenient to adjust the change rate of the weight of each layer.

Intra-layer Equalization Normalization. For the sake of clarity, the pseudo-code of the intra-layer equalization normalization is shown in Algorithm 1. For all hidden layers $k \in \{1, 2, \cdots, N\}$ in the network, we calculate the mean and standard deviation of all output values, and make all the output values satisfy the unit mean β and unit variance, that is, calculate the average value $\hat{\mu}_k(i)$ and the variance value $\hat{\sigma}_k(i)^2$ of the output value of each channel $z_k(i)$ first, and then the weights W_k and offsets b_k are divided by the coefficients respectively to make adjustment.

$$W_k(i,:) \leftarrow W_k(i,:)/\hat{\sigma}_k(i) \tag{9}$$

$$b_k(i) \leftarrow \beta = \hat{\mu}_k(i)/\hat{\sigma}_k(i) \tag{10}$$

Inter-layer Equalization Normalization. The parameter adjustment method in Intra-layer Equalization Normalization makes the output of each layer satisfy a variance of 1, and for all the rate of change $C_{k,i}^2$ in W_k is a constant. However, it does not guarantee the rate of change between layers. Here we use an iterative method to make the rate of change $C_{k,i}^2$ between all layers be a constant. The pseudo-code of crossover operator is shown in Algorithm 2.

Algorithm 1 Intra-layer Equalization Normalization

Input: the number of layers N
Output: the weights W_k and offsets b_k
1: Randomly initializes weights W_k
2: Set all the offsets as b_k=0
3: Select a part of the sample data $z_0 \in \overline{D} \subset D$ from the training data set;
4: **for all** $z_0 \in \overline{D}$ **do**
5: Obtain the output of each layer, calculate the mean $\hat{\mu}_k(i)$ and variance $\hat{\sigma}_k^2(i)$ of each channel of the output;
6: Update the scale of the weight $W_k(i,:)$ based on Equation (9)
7: Update the value of offset $b_k(i)$ based on Equation (10)
8: **end for**
9: **return** W_k, b_k

Algorithm 2. Inter-layer Equalization Normalization

Input: the number of layers N
Output: the weights W_k and offsets b_k
1: Randomly initializes weights W_k
2: Set all the offsets as b_k=0
3: Select a part of the sample data $z_0 \in \overline{D} \subset D$ from the training data set;
4: **for all** N layers **do**
5: Calculate the ratio $\overline{C}_k = E_j\left[\overline{C}_{k,j}\right]$ for each layer;
6: Calculate a scale change coefficient $r_k = \left(\bar{C}/\bar{C}_k\right)^{\alpha/2}$, α is an attenuation factor;
7: Update the weight and offset of each layer: $W_k \leftarrow r_k W_k$ and $b_k \leftarrow r_k b_k$£
8: **end for**
9: **return** W_k, b_k

3.4 Convolution Kernel Initialization Procedure Based on 2DPCA and Equalization Normalization

Assume that there are a total of n categories of picture data to be trained, the size of the convolution kernel of each convolution layer k is $R_k \times R_k$, the number of convolution kernels per layer is p_i, and N is the number of convolutional layers which determined by the model:

(1) Get the evaluation matrix set M according to the 2DPCA initialization process in Sect. 3.2;
(2) If the number of sets M is greater than or equal to the number of convolution kernels p_k, initialize all the p_k convolution kernels by taking the matrix corresponding to the previous p_k eigenvalue according to the value size λ^*; If the number of feature vectors d_k is less than the number of convolution kernels p_k, initialize the first d_k convolution kernels with all the assignment matrices, and the remaining uninitialized convolution kernel roulette algorithm randomly selects the assignment matrix into the weight matrix W_k of the current convolution layer;

(3) Using Algorithm 1, each convolutional layer is calculated separately, and calculate the mean $\hat{\mu}_k$ and variance $\hat{\sigma}_k^2$ of p_k convolution kernels within each layer;
(4) Using Algorithm 2, the entire set of weight matrixes are firstly extracted in the entire network model. The dimensions of each weight are different. The number of columns and rows in the matrix are the size and the number of the k-th layer convolution kernel, respectively. Then the iterative operation of the weight adjustment is performed:

(4.1) Traverse all N weight matrices and calculate $\overline{C}_k$ for each layer;
(4.2) Calculate the global average ratio $\bar{C}$;
(4.3) Then calculate the scale factor r_k;
(4.4) Adjust the weight W_k and bias b_k.

The above operations are iterated until the weight adjustment approaches convergence, and a new set of weight matrices $\{W'_1, W'_2, \cdots, W'_N\}$ is obtained and assigned (approximately 10 iterations).

4 Experiments and Results

The experiment of this article is trained on Alibaba Cloud Server. The CPU model is Intel Xeon Platinum 8163 (dual core). The processor is clocked at 2.5 GHz. It uses only the CPU for training. The operating system is Windows server 2012R, and the memory size is 8 GB. The programming language is c/c++.

4.1 Histogram Analysis

The output values of six hidden layers are counted in turn. The activation function uses the $Tanh$ and $Relu$ functions to draw the distribution histograms of the output values. Their histograms respectively as shown in Fig. 1 and Fig. 2, respectively.

From Fig. 1 and Fig. 2, it is relatively reasonable for the distribution of each layer of the two models. The distribution of the output values of each layer is not much different, and the parameters of back propagation adjustment are all better, which makes the model easier to training. At the same time, since the algorithm is based on the specific training sample data after the principal component analysis is performed for initialization, the initial value of the convolution kernel is more suitable for this sample data, and there will be a good starting point for the optimization of such nonconvex functions.

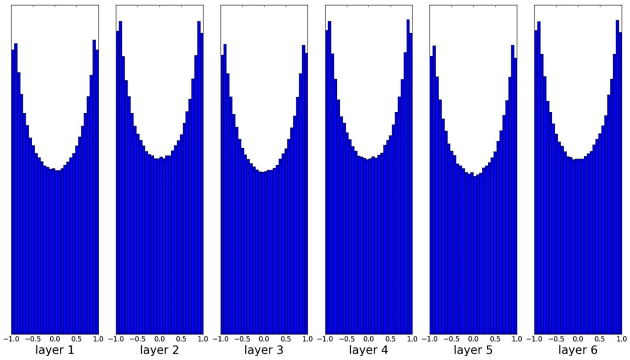

Fig. 1. The output distribution histogram with *Tanh* activation function.

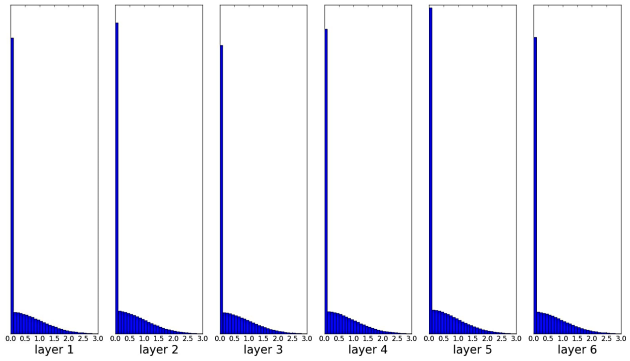

Fig. 2. The output distribution histogram with *Relu* activation function.

4.2 Model Comparison

In order to verify the performance of the initialization algorithm proposed in this chapter, the algorithm was applied to the classical hand-written digital data set MNIST for verification. *Relu* and *Tanh* were used as activation functions to calculate the accuracy rate after each training. The experimental results were compared with other classical Gaussian random initialization, Xavier initialization, and MSRA initialization methods to detect the rate of increase in recognition rate. Draw a graph of the accuracy rate of the training process.

The experiment adopts the classical Lenet-5 convolutional neural network model. The model consists of two convolutional layers and two alternating pooling layers, finally connects to a fully connected layer. Among them, the convolution kernel size is 5×5, the step length is 1, the pooling window size is 2×2, and the step size is also 2. That is, using the non-overlapping pooling and the pooling method is the maximum pooling. Whats more, the dropout regularization mechanism is added in the full connection layer to enhance the generalization ability of the network. The batch size is 1, that is, every time an image is passed

in, it will adjust weight values by back-propagating and calculate the error. In order to make comparison images more clearly and avoid the interference caused by the curve crossover, calculate the overall accuracy rate after every 10 training charts, that is, the accuracy rate equels the correct number of identification pictures/the number of trained pictures. The total number of iterations is 400, that is, the curve obtained by training 4000 images. Each algorithm counts 5 times and takes the average value.

When the activation function takes *Relu*, the comparison curve of the accuracy during the training of the three different initialization methods is shown in Fig. 3.

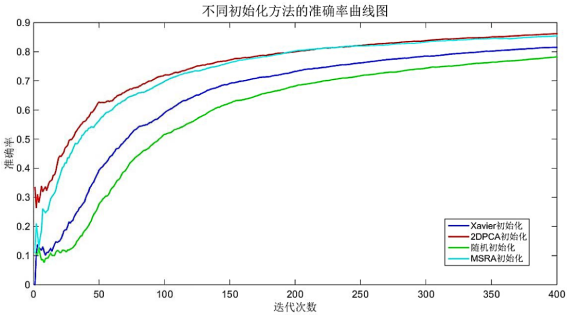

Fig. 3. The accuracy curves of different initialization methods with *Relu* activation function.

This experiment mainly contrasts the rising rate of recognition rate during training, that is, the convergence speed of the network model. From this experimental comparison chart, it can be seen that the accuracy of the four initialization methods rises as the training progresses. The recognition rate of the 2DPCA initialization method proposed in this paper rises fastest in the early stage, and it can reach the accuracy rate of 0.3 at the beginning of the training. However, the accuracy of the other methods is only 0.1 at the beginning. And for the 10 classification problem, the probability of random selection is 0.1. It can be seen that since the 2DPCA initialization is based on the actual data image initialization, a weight value suitable for the sample data can be initialized directly, and the optimization is started from a good starting point, and the accuracy rate is increased faster. And the other three methods are not based on sample data. The MSRA initialization effect is relatively good, it proves that it is indeed suitable for *Relu* activation function (Fig. 4).

It can be seen from this experiment that MSRA and Xavier have similar effects, when Tanh is used as an activation function. Since MSRA is not particularly suitable for *Tanh* functions, the effect is lower than *Relu* as an activation function, and the Xavier effect on the Tanh function is slightly higher than the MSRA initialization. Among them, the effect of 2DPCA initialized based on the sample data is still the best, and the accuracy rate of the initial period is the

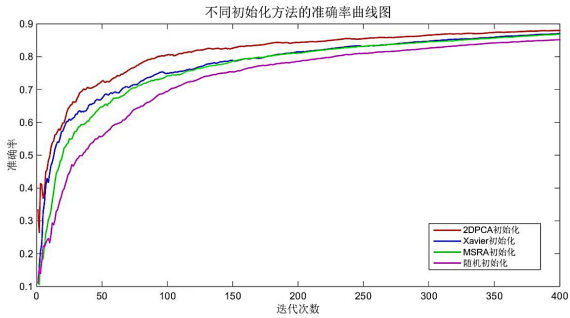

Fig. 4. The accuracy curves of different initialization methods with *Tanh* activation function.

fastest. It can be seen that the 2DPCA initialization method both has a good effect in the *Relu* activation function and the Tanh activation function, and is not limited by the type of the specific activation function.

5 Conclusion

This paper studies the initialization method of convolutional neural network, statistics the output value of each layer under different initialization methods and draws a histogram, and analyzes the distribution of output values from the histogram. 2DPCA-based convolution kernel initialization method is proposed for the problem of its distribution. 2DPCA is used to extract key features from the sample data and initialize the convolution kernel, and an equalization normalization method is introduced to adjust the size of the weights between the layers. The method does not need to manually set hyper-parameters, avoids random values, and does not have limitations on the types of activation functions. The parameters are completely determined according to the characteristics of specific training sample data. Finally, the histogram distribution and the curve comparison diagram of the model training show that the proposed method can effectively avoid the uncertainty caused by initializing the weights and accelerate the training speed of the entire model.

Acknowledgements. This work is supported by the National Key Research and Development Project (No. 2016YFC0401607), the Fundamental Research Funds for the Central Universities (No. 2019B22314), and the National Key R&D Program of China (No. 2018YFC0407101).

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034 (2015)

2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
3. Liu, X., Wang, J., Xu, K.: Novel image feature extraction algorithm based on fusion autoencoder and CNN. Appl. Res. Comput. **34**(12), 3839–3843 (2017)
4. Pacheco, A.G., Krohling, R.A., da Silva, C.A.: Restricted boltzmann machine to determine the input weights for extreme learning machines. Expert Syst. Appl. **96**, 77–85 (2018)
5. Seyfioğlu, M.S., Gürbüz, S.Z.: Deep neural network initialization methods for micro-doppler classification with low training sample support. IEEE Geosci. Remote Sens. Lett. **14**(12), 2462–2466 (2017)
6. Sun, W., Su, F., Wang, L.: Improving deep neural networks with multi-layer maxout networks and a novel initialization method. Neurocomputing **278**, 34–40 (2018)
7. Tang, J., Wang, D., Zhang, Z., He, L., Xin, J., Xu, Y.: Weed identification based on k-means feature learning combined with convolutional neural network. Comput. Electron. Agric. **135**, 63–70 (2017)
8. Thimm, G., Fiesler, E.: Neural network initialization. In: Mira, J., Sandoval, F. (eds.) IWANN 1995. LNCS, vol. 930, pp. 535–542. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59497-3_220
9. Yang, N., Li, Y., Yang, Y., Zhu, M.: Convolutional neural networks based on sparse coding for human postures recognition. In: AOPC 2017: Optical Sensing and Imaging Technology and Applications, vol. 10462, p. 104622B. International Society for Optics and Photonics (2017)
10. Yunfei, L., Randi, F., Wei, J., Nian, J.: Image super-resolution using multi-channel convolution. J. Image Graphics **22**(12), 1690–1700 (2017)
11. Zhang, Z., Ji, J.: Classification method of FMRI data based on convolutional neural network. Pattern Recog. Artif. Intell. **30**(6), 549–558 (2017)
12. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. pp. 249–256 (2010)