Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **200X**

doi:10.3934/xx.xx.xx.xx

pp. **X–XX**

A NEW INITIALIZATION METHOD BASED ON NORMED STATISTICAL SPACES IN DEEP NETWORKS

Hongfei Yang

Department of Mathematics, Yeung Kin Man Academic Building
City University of Hong Kong
Tat Chee Avenue, Kowloon Tong, Hong Kong

Xiaofeng Ding

Department of Mathematics, School of Science, Shanghai University
Shanghai 200444, P. R. China

Raymond Chan

Department of Mathematics, Yeung Kin Man Academic Building
City University of Hong Kong
Tat Chee Avenue, Kowloon Tong, Hong Kong

Hui Hu

HISILICON Technologies Co., Ltd., Huawei Base, Bantian
Longgang District, Shenzhen 518129, P. R. China.

Yaxin Peng

Department of Mathematics, School of Science, Shanghai University
Shanghai 200444, P. R. China

Tieyong Zeng*

Department of Mathematics, The Chinese University of Hong Kong
Shatin, Hong Kong

Abstract. Training deep neural networks can be difficult. For classical neural networks, the initialization method by Xavier and Yoshua which is later generalized by He, Zhang, Ren and Sun can facilitate stable training. However, with the recent development of new layer types, we find that the above mentioned initialization methods may fail to lead to successful training. Based on these two methods, we will propose a new initialization by studying the parameter space of a network. Our principal is to put constrains on the growth of parameters in different layers in a consistent way. In order to do so, we introduce a norm to the parameter space and use this norm to measure the growth of parameters. Our new method is suitable for a wide range of layer types, especially for layers with parameter-sharing weight matrices.

2010 *Mathematics Subject Classification.* Primary: 68T01, 68T05; Secondary: 68Q32.

Key words and phrases. Neural networks, parameters sharing, parameters initialization, deep learning, model training.

Raymond Chan's research is supported by HKRGC Grants No. CUHK 14306316 and CUHK 14301718, CityU Grant 9380101, CRF Grant C1007-15G, AoE/M-05/12. Tieyong Zeng's research is supported by National Science Foundation of China No. 11671002, CUHK start-up and CUHK DAG 4053342, RGC 14300219, and NSFC/RGC N_CUHK 415/19.

*Corresponding author: Tieyong Zeng (zeng@math.cuhk.edu.hk).

1. **Introduction.**

In recent years, deep learning methods have achieved remarkable progresses in image processing tasks including image restoration [24], image segmentation [14] and image super-resolution [4]. However, deep neural networks can be very difficult to train. If learnable parameters are not properly initialized, deep neural networks usually converge slowly and can get stuck to local minima easily [5]. In [1] Bradley found that if initialized randomly, the variances of gradients become smaller when they are propagated in a deep network. In the seminal work of Xavier and Yoshua [5], the authors proposed a novel initialization method based on a statistical analysis of forward and backward propagations. The Xavier initialization has the following advantages: (i) distributions of activations among different layers are approximately the same; (ii) compared to an arbitrary initialization, average ratio of the singular values of the Jacobian matrix associated with back-propagated gradients is closer to 1; and (iii) variances of back-propagated gradients are approximately the same across all layers. Later in [7] He, Zhang, Ren and Sun extended the Xavier initialization to convolution layers and the ReLU activation function. The Xavier and the He initializations are now widely used in training of deep neural networks and they are integrated in machine learning packages like TensorFlow, Keras and PyTorch. There are other popular initialization methods, see [13, 18, 21, 22].

Because the Xavier and the He initializations differ by a constant, we regard them as one method and call it the *Xavier/He initialization*. In their original papers, the Xavier/He initialization is only deduced for standard fully connected and convolution layers. By standard we mean that each entry in the weight matrix of a fully connected layer or in the kernel tensor of a convolution layer represents an individual learnable parameter. However, from experiments we find that the Xavier/He initialization may fail to lead to stable training for networks with structured layers, where weight sharing and sparsity cause uneven learning speed of parameters. This observation motivates us to find a new initialization.

In this paper we will propose a new initialization based on the Xavier/He initialization. We will introduce a norm on parameter space, and use this norm to give constrains on updates of parameters. We will give detailed calculations for various layer types for our new initialization. We will also give comparisons to the Xavier/He initialization. Our current work is a continuation of our work [3] on initialization. The significance of our new method will be explained in subsection 2.3, after we introduce necessary notations and preliminaries.

This paper is organized as the following. In the next section, we briefly give notations and preliminaries used in this paper, and we also introduce the Xavier/He initialization and the CirCNN implementation. In the third section we propose our new method and we give detailed calculations for some special cases. Then we provide some numerical experiments to demonstrate the effectiveness of our initialization.

2. **Notations and preliminaries.**

In this section we give the notations and preliminaries used in this paper.

2.1. **Single layer structure.**

Mathematically, convolution and fully connected layers can be written as

$$y = \mathrm{act}(W * x + b), \tag{1}$$

where x is the input vector, b is the bias vector, act is the activation function, and y is the output vector. For a fully connected layer, W is a matrix and $W * x$ is the matrix-vector multiplication, while for a convolution layer, W is a collection of kernels and $W * x$ is the convolution (or cross-correlation) operation (for details see [6]). We note that for both cases, $W * x$ is a linear operation on x. Therefore, after some reshaping, both layers can be rewritten as

$$y = \mathrm{act}(Wx + b), \tag{2}$$

where now W is the matrix representation of the corresponding linear operator, and Wx is the matrix vector multiplication. For a fully connected layer W is a dense matrix whose entries are learnable parameters. On the other hand, for a convolution layer W is usually a sparse matrix whose non-zero entries are built from the kernels, and the learnable parameters are the elements in the kernels.

We note that the single layer structure in (2) can represent a wide range of layer types used in deep neural networks. The difference in different layer types is how one build the matrix W from learnable parameters. Therefore in this paper we stick to (2) as the standard single layer structure, with additional information on how to build W included when necessary.

In the training process, a 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 a layer of the form (2). 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 a layer, we use $z^{(0)}$ to denote its value after initialization, and we use $z^{(1)}$ to denote its value after the first forward and backward propagation. We introduce a definition here.

Definition 2.1. Let z be a scalar variable in (2), then the *increment of* z, denoted as Δz, is defined to be

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

when we set the learning rate $r = 1$. If z is a matrix or a vector, Δ operates on z componentwisely.

For a learnable variable v we have $\Delta v = -\partial L/\partial v$. For an intermediate variable W_{nm} which equals the learnable variable v, if we assume that v only appears in W in this layer, we have

$$\Delta W_{nm} = W_{nm}^{(1)} - W_{nm}^{(0)} = v^{(1)} - v^{(0)} = -\frac{\partial L}{\partial v} = -\sum_{(n',m')} \frac{\partial L}{\partial W_{n'm'}}, \tag{3}$$

where the last summation is over all W entries that share the same parameter v.

We use $\mathbb{V}$ to denote variance of random variables. For a layer of form (2), the Xavier/He initialization conditions are specified by the followings.

Xavier/He Initialization. *Suppose the following two conditions hold:*

- *Entries in x are identically distributed with distribution $\mathcal{X}$ with mean* 0*;*
- *Entries in $\partial L/\partial y$ are identically distributed with distribution $\partial\mathcal{L}/\partial\mathcal{Y}$ with mean* 0*.*

The learnable parameters in W should be independent and identically distributed with mean 0, and the variance should be determined by the following conditions:

1. *entries in y should be identically distributed with distribution $\mathcal{Y}$ with mean 0;*
2. *entries in $\partial L/\partial x$ should be identically distributed with distribution $\partial\mathcal{L}/\partial\mathcal{X}$ with mean 0;*
3. *$\mathbb{V}(\mathcal{Y}) = \mathbb{V}(\mathcal{X})$, and $\mathbb{V}(\partial\mathcal{L}/\partial\mathcal{Y}) = \mathbb{V}(\partial\mathcal{L}/\partial\mathcal{X})$.*

If one uses ReLU as the activation function, for intermediate layers the mean of x will be positive. This can be handled by considering symmetry in W and introduce an extra scalar "gain" in the initialization as explained in [7].

2.2. **CirCNN networks.** CirCNN implementation of neural networks in introduced in [2] is a promising approach to reduce number of parameters while preserving networks' topologies. This is done by replacing matrices and convolution kernels in a neural network by block circulant matrices and block circulant convolution kernels. We say a matrix is a *circulant matrix* if it has the form

$$\begin{bmatrix} a_0 & a_1 & \cdots & a_{n-2} & a_{n-1} \\ a_{n-1} & a_0 & \cdots & a_{n-3} & a_{n-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_1 & a_2 & \cdots & a_{n-1} & a_0 \end{bmatrix}. \tag{4}$$

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 form (2), we say W is a *block-circulant matrix* if W consists of block matrices, where each block is a circulant matrix. For a convolution layer with kernel $W = (W_{ijk\ell})$, where the k and ℓ indices represent the input and output channels and the i and j indices represent $2D$ kernels, we say W is *block-circulant* if for each fixed i, j the resulting matrix $(W_{ijk\ell})_{k\ell}$ is block-circulant. As suggested in [2], for a single layer the block size should be a constant, while one can choose different block sizes for different layers.

Using CirCNN implementation can significantly reduce number of learnable parameters. If we use B as the block size, then for a fully connected layer or a convolution layer we can reduce learnable parameters in W by B times. We shall call B the *compression ratio*. However, from experiments we observe that when using large compression ratio, the network tends to be difficult to train even if we employ the Xavier/He initialization.

Assume that we have a fully connected layer $y = \text{act}(Wx + b)$ with input and output dimensions equal to n, where W is a circulant matrix of form (4). The Xavier/He initialization leads to the following: entries in the first row of W, which are the learnable parameters that build up W, should have mean 0 and variance gain/n. Here gain is a scalar determined by activation function, e.g. $\text{gain} = \sqrt{2}$ for ReLU activation and $\text{gain} = \sqrt{2/(1+\ell^2)}$ for Leaky ReLU activation with negative slope ℓ. We have

$$\Delta W_{\ell i} = \Delta a_{[i-\ell]} = -\sum_{j=0}^{n-1} \frac{\partial L}{\partial W_{j,[i-\ell+j]}}, \tag{5}$$

where $[i]$ means $i \mod n$. This shows that in general, on the scale level, $\Delta W_{\ell i}$ does not equal to $\partial L/\partial W_{\ell i}$. This is in contrast to a standard fully connected layer, where $\Delta W_{\ell i} = -\partial L/\partial W_{\ell i}$. Therefore we intend to introduce a metric on the parameter space, and to introduce a new initialization method, so that we can preserve the desired relationship among the scales of ΔW and $\partial L/\partial W$.

2.3. **Comparison with our previous approach.** In [3] we have proposed an initialization which gives fast and stable training for networks with CirCNN implemented fully connected layers. In this subsection we explain the significance of the initialization in this paper compared to our previous approach.

Firstly, the initialization in [3] is only for fully connected layers, and for convolution layers we have to rely on the Xavier/He initialization. This is because for a convolution layer, the matrix W in (2) is a sparse matrix. The zero elements in W are not learnable parameters and in [3] we do not know how to deal with non-learnable entries. On the other hand, the initialization that will be introduced in this paper can handle both fully connected and convolution layers. From experiments we observe that for convolution layers, the Xavier/He initialization performs well only for small kernels. Our new method in this paper performs well on both small and large kernels. See Table 2 and Figure 1(b) for a numerical demonstration. Secondly, the initialization condition (equation (8)) in [3] is based on an analysis for individual entries in the weight matrix W. This makes it difficult to generalize the initialization method in [3]. On the other hand, our analysis in this paper is based on the whole matrix W, and we will introduce a norm to put constraints on true increments of W based on partial derivatives of the loss with respect to W. This not only gives more geometric flavour to our condition (equality in the length of two vectorized matrices as in (20)), but it also gives a unified framework to deduce initializations for various layer types.

3. **A new initialization method.** Let us assume that each layer in a neural network can be written in the following form

$$\begin{cases} \bar{w} = (c \cdot T)v, \\ W_{nm} = \bar{w}_{(nm)}, \\ y = \mathrm{act}(Wx + b), \end{cases} \tag{6}$$

where $x \in \mathbb{R}^M$, $y \in \mathbb{R}^N$, $W \in \mathbb{R}^{N\times M}$ and $b \in \mathbb{R}^N$. We vectorize W as a column vector $\bar{w}$ with the indexing correspondence $(nm) = (n-1)M + m$. The vector $v \in \mathbb{R}^\Lambda$ contains learnable parameters. The matrix $T \in \mathbb{R}^{NM\times\Lambda}$ builds $\bar{w}$ from v, and $c \in \mathbb{R}$ is a positive scalar that we determine later. We stress that in (6) the learnable parameters are contained in v and b, and in a backpropagation v and b will be updated, but not W. As in previous analysis of initializations in [5, 7], we regard elements in each terms in (6) as random variables. For Xavier/He initialization or the methods in [13, 18, 22] we always have $c = 1$.

For initialization of layer (6), we need to determine the parameter c and a variance, such that entries in v are independent and identically distributed with mean 0 and the variance. In practice one can use Gaussian or uniform distribution. We always use uniform distribution in our numerical demonstrations.

Let us use L to denote a loss function. Then our initialization method is defined below.

Our Initialization. *Suppose the following conditions hold:*

- *Entries in x are identically distributed with distribution $\mathcal{X}$ with mean* 0*;*
- *elements in each row or in each column of W are independent;*

- *and entries in $\partial L/\partial y$ are identically distributed with distribution $\partial\mathcal{L}/\partial\mathcal{Y}$ with mean 0.*

Entries in v are initialized as independent and identically distributed random variables with mean 0. The variance of v and the parameter c should be determined by the following :

1. *Entries in y are identically distributed with distribution $\mathcal{Y}$ with mean 0;*
2. *entries in $\partial L/\partial x$ are identically distributed with distribution $\partial\mathcal{L}/\partial\mathcal{X}$ with mean 0;*
3. *$\mathbb{V}(\mathcal{Y}) = \mathbb{V}(\mathcal{X})$ and $\mathbb{V}(\partial\mathcal{L}/\partial\mathcal{Y}) = \mathbb{V}(\partial\mathcal{L}/\partial\mathcal{X})$;*
4. *and $\mathbb{E}(||\Delta\bar{w}||_2^2) = \mathbb{E}(||\partial L/\partial\bar{w}||_2^2)$.*

Note that conditions 1 to 3 are the adaptations of the Xavier/He initialization conditions to the layer type (6), while the condition 4 is a probabilistic description of a norm equality, which is explained at the end of this section. The assumption on rows and columns of W is a mild assumption that is satisfied by many layers, like fully connected layers, convolution layers, their CirCNN implementations, depthwise convolutions [9], interleaved group convolutions [25] and dilated convolutions [23]. We note that like the Xavier/He initialization, when one uses the ReLU activation function, entries in y are non-negative, and thus $\mathcal{Y}$ cannot have mean 0. We use the same technique as in [7] to introduce a constant scaler "gain" to tackle this difficulty.

In order to give detailed calculations of initializations, in the rest of this paper we assume that the matrix T in (6) has the following conditions: entries in T are either 0 or 1; each row of T has at most one 1; and the numbers of 1's in different columns of T are all the same. We also assume that the vector v in (6) only contains learnable parameters for this layer, which simplifies the calculations. We note that weight sharing across layers can also be handled by our new initialization, see Example 4.2 in [3] for a demonstration for the fully connected case.

We denote the number of non-zero entries in each column of T as K. We calculate as follows. For each n, m we have

$$\begin{aligned}\left(\Delta\bar{w}_{(nm)}\right)^2 &= \left(c\sum_{\lambda} T_{(nm),\lambda}\Delta v_\lambda\right)^2 \\ &= c^2\sum_{\lambda} T_{(nm),\lambda}^2\left(\Delta v_\lambda\right)^2 + c^2\sum_{\lambda_1\neq\lambda_2} T_{(nm),\lambda_1}T_{(nm),\lambda_2}\Delta v_{\lambda_1}\Delta v_{\lambda_2} \\ &= c^2\sum_{\lambda} T_{(nm),\lambda}^2\left(\Delta v_\lambda\right)^2,\end{aligned} \tag{7}$$

where the last equality comes from the fact that in each row of T there is at most one non-zero element. Therefore taking expectation on the squared 2-norm gives

$$\begin{aligned}\mathbb{E}(||\Delta\bar{w}||_2^2) &= \mathbb{E}\left(c^2\sum_{\lambda,n,m} T_{(nm),\lambda}^2\left(\Delta v_\lambda\right)^2\right) \\ &= c^2\sum_{\lambda,n,m} T_{(nm),\lambda}^2\mathbb{E}\left(\left(\Delta v_\lambda\right)^2\right) \\ &= c^2\sum_{\lambda,n,m} T_{(nm),\lambda}^2\mathbb{V}(\Delta v_\lambda).\end{aligned} \tag{8}$$

We have

$$\Delta v_\lambda = -\partial L/\partial v_\lambda = -c\sum \partial L/\partial \bar{w}_{ij}, \tag{9}$$

where the last summation is over all entries in $\bar{w}$ that correspond to v_λ, and there are K terms in the summation. Here we make an assumption that the random variables $\{\partial L/\partial \bar{w}_{ij}\}$, where $\bar{w}_{ij}$ corresponds to one learnable parameter v_λ, are uncorrelated and identically distributed with variance $\mathbb{V}(\partial\mathcal{L}/\partial\mathcal{W})$. Then we have

$$\mathbb{V}(\Delta v_\lambda) = c^2\mathbb{V}(\sum \partial L/\partial \bar{w}_{ij}) = c^2 K \cdot \mathbb{V}(\partial\mathcal{L}/\partial\mathcal{W}). \tag{10}$$

Combining (8) and (10) we have

$$\mathbb{E}(||\Delta\bar{w}||_2^2) = c^4 K|T|\mathbb{V}(\partial\mathcal{L}/\partial\mathcal{W}), \tag{11}$$

where $|T|$ denotes the number of non-zero entries in T. On the other hand it is easy to see that

$$\mathbb{E}(||\partial L/\partial\bar{w}||_2^2) = NM\mathbb{V}(\partial\mathcal{L}/\partial\mathcal{W}). \tag{12}$$

By combining (11) and (12) we conclude that the scalar value c should be

$$c = \sqrt[4]{NM/(K|T|)}. \tag{13}$$

After c is calculated, the variance $\mathbb{V}(v)$ is determined by the standard Xavier/He initialization as described in [5, 7]. Let L_N, L_M denote the number of non-zero entries in rows and columns of W respectively. Then we have

$$\mathbb{V}(v) = \frac{\text{gain}\cdot 2}{c^2\cdot(L_N + L_M)}, \tag{14}$$

where gain is a scalar determined by the activation function.

To sum it up, our new initialization method consists of 3 steps. The first step is to write a layer in the form (6). The second step is to calculate c and $\mathbb{V}(v)$ by (13) and (14). The last step is to initialize entries in v by independent and identically distributed variables with mean 0 and variance $\mathbb{V}(v)$, and initialize the bias vector b as the zero vector.

To illustrate our new initialization method, in the rest of this section we consider two cases: CirCNN implementations of fully connected layers with block-size B, and CirCNN implementations of 2-D convolution layers with block-size B. Note that when $B = 1$, these two cases correspond to standard fully connected layers and standard 2-D convolution layers.

For CirCNN implementations of fully connected layers with kernel size B, v is a vector of size NM/B. Therefore $K = B$ and $|T| = NM$, and so $c = \sqrt[4]{NM/(K|T|)} = 1/\sqrt[4]{B}$. The matrix W is a dense matrix with $L_N = N$ and $L_M = M$, so

$$\begin{aligned} \mathbb{V}(v) &= \frac{\text{gain}\cdot 2}{c^2\cdot(L_N + L_M)} \\ &= \frac{\text{gain}\cdot 2\sqrt{B}}{M+N}. \end{aligned} \tag{15}$$

This gives the same initialization as in [3] for fully connected layers. For standard fully connected layers (i.e. $B = 1$), our initialization gives same initialization with the Xavier/He initialization.

For CirCNN 2-D convolution layers with block-size B, suppose the 2-D kernel size is r (for a 3×3 kernel, $r = 9$). Suppose the input to this layer is a 3-D tensor of size $M'\times H\times W$, M' is the input channel number, and the output channel size is N'.

Then v is a vector of size $N'M'r/B$. We have $N = N'HW$, $M = M'HW$, $|T| = N'HWrM'$, $K = HWB$, $L_N = N'r$ and $L_M = M'r$. Therefore $c = \sqrt[4]{1/(rB)}$, and

$$\mathbb{V}(v) = \frac{\text{gain} \cdot 2\sqrt{B}}{\sqrt{r}(M' + N')}. \tag{16}$$

For standard 2-D convolution layers (i.e. $B = 1$), the Xavier/He initialization gives $c = 1$ and

$$\mathbb{V}(v) = \frac{\text{gain} \cdot 2}{r(M' + N')}, \tag{17}$$

which is different from our initialization.

The new condition 4 in our initialization is based on a normed statistical space that we briefly describe here. Let S^N denote the collection of N-dimensional random variables. For $x, y \in S^N$, we say $x \sim y$ if there exits a constant vector γ such that $P(x - y = \gamma) = 1$. Then $\sim$ defines an equivalent relationship, and for $x \in S^N$ we denote its equivalent class by $[x]$. The collection of all equivalent classes is denoted by H^N. For any $[x], [y] \in H^N$ and $\alpha \in \mathbb{R}$, it can be shown that the two definitions $[x] + [y] := [x + y]$ and $\alpha[x] := [\alpha x]$ are well defined, and they turn H^N into a vector space. The crucial observation is that we can define an inner product on H^N, which is given by

$$\langle [x], [y] \rangle = \sum_{i=1}^{N} \mathbb{E}\left((x_i - \mathbb{E}(x_i))(y_i - \mathbb{E}(y_i))\right). \tag{18}$$

Note that when $N = 1$, the above inner product is just the covariance of the random variables x and y. This construction turns our vector space H^N into a normed space $(H^N, \|\cdot\|)$, where the norm is defined by

$$\|[x]\| = \sqrt{\langle [x], [x] \rangle}. \tag{19}$$

Therefore our new condition 4 in our initialization is the same as

$$\|\Delta\bar{w}\| = \|\partial L/\partial \bar{w}\|. \tag{20}$$

This gives a new explanation of the condition 4: the true increment $\Delta\bar{w}$ has the same length as the length of the partial derivative vector $\partial L/\partial\bar{w}$ under the norm in (19). We note that the above is a standard construction, where one quotients an inner product space by a closed subspace [15, 16].

3.1. **Numerical experiments.** Here we present experiments to demonstrate the effectiveness of our method. For each comparison all hyper-parameters are set to the same except learning rates. For the experiment with network structure in Table 1, the learning rate for our initialization is set to $\sqrt{3}$ times the learning rate for the Xavier/He initialization (note that for convolution layers, $c = \sqrt[4]{9} = \sqrt{3}$). For other experiments, we will fix a learning rate and run training 5 times for each initialization. The learning rates are tuned by a factor of 2 as the following: we select the learning rate such that in all 5 runs the losses exhibit the fastest decreasing, with no loss stalls or explodes. We always use SGD with no momentum as optimizers. We will report the mean and standard deviation (std) of the five runs. For losses in Figures 1(a)–(d) we first apply a 0.99 smoothing in TensorBoard, and we report the mean and std of last value of the smoothed losses.

We test our initialization on various networks as summarized in Tables 1–4, with the corresponding plots of losses in Figures 1(a)–(d). Here we use the MNIST data set [12] for training. We note that for the simple network summarized in Table 1, where the kernels of convolutions are all small and there is no CirCNN implementation, our initialization behaves almost identical to the Xavier/He initialization. However, as illustrated by Figure 1(b), when the network has a convolution layer with very large kernels, our initialization outperforms the Xavier/He initialization. Plots in Figures 1(c) and (d) show the effectiveness of our initialization for CirCNN implementations.

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

TABLE 1. Network structure of Figure 1(a).

Layer	Output channel	Number of parameters
Conv2d+Maxpool	32	$3 \times 3 \times 1 \times 32$
Conv2d+Maxpool	64	$3 \times 3 \times 32 \times 64$
Reshape to 56×56		
Conv2d	1	$55 \times 55 \times 1 \times 1$
Fc	10	3136×10

TABLE 2. Network structure of Figure 1(b). For the last convolution layer with kernel size 55×55 we use periodic padding on the input images to make sure the conditions on T in (6) are satisfied.

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

TABLE 3. Network structure of Figure 1(c). For a compression ratio $B > 1$, we use circulant implementation with block size B. The number of parameters for a CirCNN implementation should be divided by B.

Lastly we test our initialization on a CirCNN implementation of the VGG16 network [19] with the Cifar10 data set [11]. For the second fully connected layer in the network which has 4096 input and output channels, we use a circulant matrix with block size 4096. The original VGG16 network (with no CirCNN) can achieve 92.24% top-1 accuracy. On 5 runs, the network with our initialization consistently outperforms the network with the Xavier/He initialization on top-1 accuracy. The average top-1 accuracy for our method is 91.38% with std 0.199, while the average top-1 accuracy for the Xavier/He initialization is 90.98% with std 0.188 (std calculated on percentile scale).

Layer	Out channel	Number of parameters	Compression ratio
Conv2d+Maxpool	32	$3 \times 3 \times 1 \times 32$	1
Conv2d+Maxpool	64	$3 \times 3 \times 32 \times 64$	1
Conv2d	256	$3 \times 3 \times 64 \times 256$	1
Conv2d	256	$3 \times 3 \times 256 \times 256$	256
Conv2d	256	$3 \times 3 \times 256 \times 256$	256
Fc	64	12544×64	1
Fc	10	64×10	1

Table 4. Network structure of Figure 1(d). For a compression ratio $B > 1$, we use circulant implementation with block size B. The number of parameters for a CirCNN implementation should be divided by B.

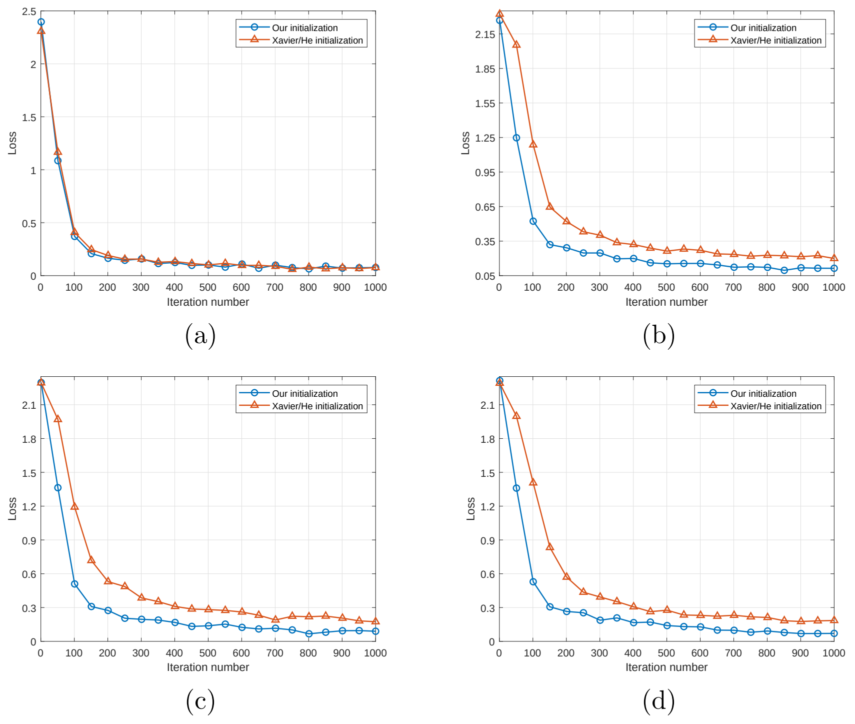

Figure 1. (a) Plot of losses of network summarized in Table 1. (b) Plot of losses of network summarized in Table 2. (c) Plot of losses of network summarized in Table 3. (d) Plot of losses of network summarized in Table 4. Mean and std for the last of the smoothed loss values: Ours (a) 0.070 ± 0.005, (b) 0.111 ± 0.006, (c) 0.088 ± 0.003 , (d) 0.083 ± 0.004; Xavier/He (a) 0.069 ± 0.001, (b) 0.206 ± 0.012, (c) 0.221 ± 0.016, (d) 0.164 ± 0.012. We also tested the evaluation accuracies on the test set with results: Ours versus Xavier/He (a) 98.06%, 98.13%, (b) 95.66%, 93.94%, (c) 97.17%, 95.07%, (d) 98.01%, 96.22%.

4. Conclusion and future works.

Based on the Xavier/He initialization, we propose a novel initialization. We introduce a norm on the parameter space, and we use this norm in our initialization to constrain growth of parameters. We focus on the true increment matrix ΔW and its relation with the partial derivatives $\partial L/\partial W$ which are not studied in previous works. Our new initialization is suitable for fully connected layers, convolution layers and many other layer types. Numerical experiments show that our initialization can lead to stable and fast training, especially for networks with heavy weight sharing.

Networks with shortcuts, like Resnets [8], Inception resnets [20], squeeze nets [10] and Mobile Net V2 [17], does not satisfy the conditions in Xavier/He initialization method. This is because when a network has shortcuts, some inputs x have non-independent elements. Therefore our current approach cannot be applied to these networks. We plan to extend our initialization to these networks by regularizing the extra gradient flows in shortcuts. One possible approach is to introduce another scaler c', like the c in (6), to adjust flows in shortcuts.

REFERENCES

[1] D. M. Bradley, *Learning in Modular Systems*, PhD thesis, Carnegie Mellon University, 2010.

[2] C. Ding, S. Liao, Y. Wang, Z. Li, N. Liu, Y. Zhuo, C. Wang, X. Qian, Y. Bai, G. 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*, ACM, 2017, 395–408.

[3] X. Ding, H. Yang, R. Chan, H. Hu, Y. Peng and T. Zeng, A new initialization method for neural networks with weight sharing, Submitted for publication.

[4] C. Dong, C. C. Loy, K. He and X. Tang, Image super-resolution using deep convolutional networks, *IEEE transactions on pattern analysis and machine intelligence*, **38** (2015), 295–307.

[5] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, 249–256.

[6] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016, http://www.deeplearningbook.org.

[7] K. He, X. Zhang, S. Ren and J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[8] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv:1704.04861

[10] J. Hu, L. Shen and G. Sun, Squeeze-and-excitation networks, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, 7132–7141.

[11] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*, Technical report, Citeseer, 2009.

[12] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, **86** (1998), 2278–2324.

[13] D. Mishkin and J. Matas, All you need is a good init, in *International Conference on Learning Representations*, 2016.

[14] O. Ronneberger, P. Fischer and T. Brox, U-net: Convolutional networks for biomedical image segmentation, in *International conference on Medical image computing and computer-assisted intervention*, 2015, 234–241.

[15] W. Rudin, *Real and complex analysis*, 3rd edition, McGraw-Hill Book Co., New York, 1987.

[16] W. Rudin, *Functional analysis*, 2nd edition, International Series in Pure and Applied Mathematics, McGraw-Hill, Inc., New York, 1991.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, 4510–4520.

[18] A. Saxe, J. L. McClelland and S. Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, arXiv:1312.6120

[19] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556

[20] C. Szegedy, S. Ioffe, V. Vanhoucke and A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[21] M. Taki, Deep residual networks and weight initialization, arXiv:1709.02956

[22] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. Schoenholz and J. 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*, 2018, 5389–5398.

[23] F. Yu and V. Koltun, Multi-scale context aggregation by dilated convolutions, arXiv:1511.07122

[24] K. Zhang, W. Zuo, S. Gu and L. Zhang, Learning deep cnn denoiser prior for image restoration, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, 3929–3938.

[25] T. Zhang, G.-J. Qi, B. Xiao and J. Wang, Interleaved group convolutions, in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, 4373–4382.

Received xxxx 20xx; revised xxxx 20xx.

E-mail address: honyang@cityu.edu.hk
E-mail address: dxfeng@shu.edu.cn
E-mail address: rchan.sci@cityu.edu.hk
E-mail address: huhui12@huawei.com
E-mail address: yaxin.peng@shu.edu.cn
E-mail address: zeng@math.cuhk.edu.hk