All You Need is Beyond a Good Init: Exploring Better Solution for Training Extremely Deep Convolutional Neural Networks with Orthonormality and Modulation

Di Xie
xiedi@hikvision.com

Jiang Xiong
xiongjiang@hikvision.com

Shiliang Pu
pushiliang@hikvision.com

Hikvision Research Institute
Hangzhou, China

Abstract

Deep neural network is difficult to train and this predicament becomes worse as the depth increases. The essence of this problem exists in the magnitude of backpropagated errors that will result in gradient vanishing or exploding phenomenon. We show that a variant of regularizer which utilizes orthonormality among different filter banks can alleviate this problem. Moreover, we design a backward error modulation mechanism based on the quasi-isometry assumption between two consecutive parametric layers. Equipped with these two ingredients, we propose several novel optimization solutions that can be utilized for training a specific-structured (repetitively triple modules of Conv-BN-ReLU) extremely deep convolutional neural network (CNN) WITHOUT *any shortcuts/ identity mappings from scratch. Experiments show that our proposed solutions can achieve distinct improvements for a 44-layer and a 110-layer plain networks on both the CIFAR-10 and ImageNet datasets. Moreover, we can successfully train plain CNNs to match the performance of the residual counterparts.*

Besides, we propose new principles for designing network structure from the insights evoked by orthonormality. Combined with residual structure, we achieve comparative performance on the ImageNet dataset.

1. Introduction

Deep convolutional neural networks have improved performance across a wider variety of computer vision tasks, especially for image classification [17, 34, 39, 31, 45], object detection [42, 26, 33] and segmentation [20, 5, 25]. Much of this improvement should give the credit to gradually deeper network architectures. In just four years, the layer number of networks escalates from several to hundreds, which learns more abstract and expressive representations from large amount of data, *e.g*. [27]. Simply stacking more layers onto current architectures is not a reasonable solution, which incurs vanishing/exploding gradients [4, 9]. To handle the relatively shallower networks, a variety of initialization and normalization methodologies are proposed [9, 30, 12, 37, 15, 22, 13, 1], while deep residual learning [11] is utilized to deal with extremely deep ones.

Though other works, *e.g*. [36, 35], have also announced that they can train an extremely deep network with improved performance, deep residual network [11] is still the best and most practical solution for dealing with the degradation of training accuracy as depth increases. However, it is substantial that residual networks are exponential ensembles of relatively shallow ones (usually only 10-34 layers deep), as an interpretation by Veit *et al*. [41], it avoids the vanishing/exploding gradient problem instead of resolving it directly. Intrinsically, the performance gain of networks is determined by its multiplicity, not the depth. So how to train an ultra-deep network is still an open research question with which few works concern. Most researches still focus on designing more complicated structures based on residual block and its variants [18, 43]. Anyway, dose there exist an applicable methodology that can be used for training a genuinely deep network?

In this paper, we try to find a direct feasible solution to answer above question. We think batch normalization (BN) [13] is necessary to ensure the propagation stability in the forward pass in ultra-deep networks and the key of learning availability exists in the backward pass which propagates errors with a top-down way. We constrain the network's structure to repetitive modules consisted by Convolution, BN and ReLU [23] layers (Fig. 1) and analyze the Jacobian of the output with respect to the input between consecutive modules. We show that BN cannot guarantee the magnitude of errors to be stable in the backward pass and this amplification/attenuation effect to signal will ac-

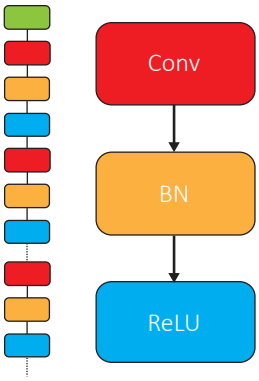

Figure 1. Diagram of the plain CNN network architecture (left) and repetitive triple-layer module (right) in this paper. Green box is for input data, Red color ones denotes parametric layers (convolutional or fully connected), yellow represents batch normalization layers and blue means activation layers. Actually, this structure is similar with the plain CNN designed by He *et al*. [11].

cumulate layer-wisely which results in gradients exploding/vanishing. From the view of norm-preserving, we find that keeping the orthonormality between filter banks within a layer during learning process is a sufficient and necessary condition to ensure the stability of backward errors. While this condition cannot be satisfied in nonlinear networks equipped with BN, this orthonormal constrain can mitigate backward signal's attenuation and we prove it by experiments. An orthonormal regularizer is introduced to replace traditional weight decay regularization [8]. Experiments show that there is $3\% \sim 4\%$ gains for a 44-layer network on CIFAR-10.

However, as depth increases, *e.g*. deeper than 100 layers, the non-orthogonal impact induced by BN, ReLU and gradients updating accumulates, which breaks the dynamic isometry [30] and makes learning unavailable. To neutralize this impact, we design a modulation mechanism based on the quasi-isometry assumption between two consecutive parametric layers. We show the quasi-isometry property with both mathematical analysis and experiments. With the modulation, a global scale factor can be applied on the magnitude of errors a little unscrupulously during the backward pass in a layer-wise fashion. Combined with orthonormality, experiments show that a plain CNN shown in Fig. 1 can be trained relatively well and match the performance of its residual counterpart.

The contributions of this paper are summarized as follows. 1) We demonstrate the necessity of applying BN and explain the potential reason which results in degradation problem in optimizing deep CNNs; 2) A concise methodology equipped with orthonormality and modulation is proposed to provide more insights to understand learning dynamics of CNNs; 3) Experiments and analysis exhibit interesting phenomenons and promising research directions.

2. Related Work

Initialization in Neural Networks. As depth increases, Gaussian initialization cannot suffice to train a network from scratch [34]. The two most prevalent works are proposed by Glorot & Bengio [9] and He *et al*. [12] respectively. The core idea of their works is to keep the unit variance of each layer's output. Sussillo & Abbott [37] propose a novel random walk initialization and mainly focus on adjusting the so-called scalar factor g to make the ratio of input/output error to be constant around 1. Krähenbühl *et al*. [15] introduce data-dependent initialization to ensure all layers training at an equal rate.

Orthogonality is also in consideration. Saxe *et al*. [29, 30] analyse the dynamics of learning in linear deep neural networks. They find that the convergence rate of random orthogonal initialization of weights is equivalent to unsupervised pre-training, which are both superior to random Gaussian initialization. LSUV initialization method [22] is proposed which not only takes advantage of orthonormality but also makes use of the unit-variance of each layer's output.

In our opinion, a well-behaved initialization is not enough to resist the variation as learning progresses, which is to say, to have a good initial condition (*e.g*. isometry) cannot ensure the preferred condition to keep unchanged all the time, especially in extremely deep networks. This argument forms the basic idea that motivates us to explore the solutions for genuinely deep networks.

Signal Propagation Normalization. Normalization is a common and ubiquitous technique in machine learning community. The whitening and decorrelation of input data brings benefits to both deep learning and other machine learning algorithms, which helps speeding up the training process [19]. Batch normalization [13] generalize this idea to ensure each layer's output to be identical distributions which reduce the internal covariate shift. Weight normalization [28] is inspired by BN by decoupling the norm of the weight vector from its direction while introducing independencies between the examples in a minibatch. To overcome the disadvantage of BN that dependent on minibatch size, layer normalization [2] is proposed to solve the normalization problem for recurrent neural networks. But this method cannot be applied to CNN, as the assumption violates the statistics of the hidden layers. For more applicable in CNN, Arpit *et al*. introduce normalization propagation [1] to reduce the internal covariate shift for convolutional layers and even rectified linear units. The idea of normalization each layers' activations is promising, but a little idealistic in practice. Since the incoherence prior of weight matrix is actually not true in the initialization phase and even worsen in iterations, the normalized magnitude of each layer's activa-

tions cannot be guaranteed in an extremely deep network. In our implementation, it even cannot prevent the exploding activations' magnitude just after initialization.

Signal Modulation. Few work is done in this field explicitly, but implicitly integrated the idea of modulation. In a broad sense, modulation can be viewed as a persistent process of the combination of normalization and other methodology to keep the magnitude of a variety of signals steady at learning. With this understanding, we can summarize all the methods above with a unified framework, *e.g*. batch normalization [13] for activation modulation, weight normalization [28] for parameter modulation, *etc*.

3. Methodology

3.1. Why is BN a requisite?

Since the complexity dynamics of learning in nonlinear neural networks [30], even a proven mathematical theory cannot guarantee that a variety of signals keeping isometrical at the same time in practice applications. Depth itself results in the "butterfly effect" with exponential diffusion while nonlinear gives rise to indefiniteness and randomness. Recently proposed methods [1, 37, 15] which utilize isometry fail to keep the steady propagation of signals in over-100-layer networks. These methods try to stabilize the magnitude of signals from one direction (forward/backward) as a substituted way to control the signals in both directions. However, since the complexity variations of signals, it is impossible to have conditions held on both ways with just one modulation method.

An alternative option is to simplify this problem to constrain the magnitude of signals in either direction, which we can pay the whole attention to another direction[1]. Batch normalization is an existed solution that satisfies our requirement. It does normalization in the forward pass to reduce internal covariate shift with a layer-wise way[2], which, in our opinion, make us to focus all the analyses on the opposite direction.

From [13], during the backpropagation of the gradient of loss ℓ through BN, we can formulate errors between adjacent layers as follow:

$$\frac{\partial\, \ell}{\partial\, x_i} = \frac{1}{\sqrt{\sigma_B^2 + \epsilon}}(\delta_i - \mu_\delta - \frac{\hat{x}_i}{m}\sum_{j=1}^{m}\delta_j \hat{x}_j) \qquad (1)$$

where x_i is ith sample in a mini-batch (we omit activation index for simplicity), so $\frac{\partial\, \ell}{\partial\, x_i}$ denotes output error. $\delta_i = \frac{\partial\, \ell}{\partial\, y_i} \cdot \gamma$ where $\frac{\partial\, \ell}{\partial\, y_i}$ is the input error and γ is scale parameter of BN. $\mu_\delta = \frac{1}{m}\sum_{i=1}^{m}\delta_i$ is mean of scaled input errors, where m denotes mini-batch's size. $\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$ is the corresponding normalized activation.

Equation 1 represents a kind of "pseudo-normalization" transformation for error signals δ_i compared with its forward operation. If the mean of distribution of input error δ_i is zero and symmetric, we can infer that the mean of distribution of output error is approximately zero. It centralizes the errors and the last term $\frac{\hat{x}_i}{m}\sum_{j=1}^{m}\delta_j \hat{x}_j$ will bias the distribution but these biases may be cancelled out from each other owing to the normalized coefficient $\hat{x}_i$ which is normal distribution. Besides, errors are normalized with a mismatched variance. This type of transformation will change error signal's original distribution with a layer-wise way since the second order moment of each layer's output errors loses its isometry progressively. However, this phenomenon can be ignored when we only consider a pair of consecutive layers. In a sense, we can think the backward propagated errors are also normalized as well as its forward pass, which is why we apply "Conv-BN-ReLU" triple instead of "Conv-ReLU-BN"[3].

The biased distribution effect will accumulated as depth increases and distort input signals' original distribution, which is one of several reasons that make training extreme deep neural network difficult. In next section we try to solve the problem to some extent.

3.2. Orthonormality

Norm-preserving resides in the core idea of this section. A vector $\mathbf{x} \in \Re^{d_\mathbf{x}}$ is mapped by a linear transformation $\mathbf{W} \in \Re^{d_\mathbf{x} \times d_\mathbf{y}}$ to another vector $\mathbf{y} \in \Re^{d_\mathbf{y}}$, say, $\mathbf{y} = \mathbf{W}^T\mathbf{x}$. If $\|\mathbf{y}\| = \|\mathbf{x}\|$, then we call this transformation norm-preserving. Obviously, orthonormality, not the normalization proposed by [1] alone, is both sufficient and necessary for holding this equation, since

$$\|\mathbf{y}\| = \sqrt{\mathbf{y}^T\mathbf{y}} = \sqrt{\mathbf{x}^T\mathbf{W}\mathbf{W}^T\mathbf{x}} = \sqrt{\mathbf{x}^T\mathbf{x}} = \|\mathbf{x}\| \; iff. \; \mathbf{W}\mathbf{W}^T = \mathbf{I} \qquad (2)$$

Given the precondition that signals in forward pass are definitely normalized, here we can analyse the magnitude variation of errors only in backward pass. To keep the gradient with respect to the input of previous layer norm-preserving, it is straightforward to conclude that we would better maintain orthonormality among columns[4] of a weight matrix in a specific layer during learning process rather than at initialization according to Eq. 2, which equivalently

[1]For a specified weight that connected ith neuron in lth layer and kth neuron in $(l+1)$th layer, $w_{ij}^{(l)}$, its gradient can be computed as $\nabla\, w_{ij}^{(l)} = a_i^{(l)} \times \delta_j^{(l+1)}$. If the two variables are independent from each other, then the magnitude of gradient can be directly related with just one factor (activation/error).

[2]Methods modulate signals without a layer-wise manner, *e.g*. [1], will accumulate the indefiniteness with a superlinear way and finally the propagated signals will be out of control.

[3]Another reason is that placing ReLU after BN guarantees approximately 50% activations to be nonzero, while the ratio may be unstable if putting it after convolution operation.

[4]Beware of the direction, which results in the exchange of notations in equation 2. So the rows and columns of the matrix are also exchanged.

makes the Jacobian to be ideally dynamical isometry [30]. Obviously in CNN this property cannot be ensured because of 1) the gradient update which makes the correlation among different columns of weights stronger as learning proceeding; 2) nonlinear operations, such as BN and ReLU, which destroy the orthonormality. However, we think it is reasonable to force the learned parameters to be conformed with the orthogonal group as possible, which can alleviate vanishing/exploding phenomenon of the magnitude of errors and the signal distortion after accumulated nonlinear transformation. The rationality of these statements and hypotheses has been proved by experiments.

To adapt the orthonormality for convolutional operations, we generalize the orthogonal expression with a direct modification. Let $\tilde{\mathbf{W}}_l \in \Re^{W\times\ H\times\ C\times\ M}$ denote a set of convolution kernels in lth layer, where W, H, C, M are width, height, input channel number and output channel number, respectively. We replace original weight decay regularizer with the orthonormal regularizer:

$$\frac{\lambda}{2}\sum_{i=1}^{D}\|\mathbf{W}_l^T\mathbf{W}_l-\mathbf{I}\|_F^2 \tag{3}$$

where λ is the regularization coefficient as weight decay, D is total number of convolutional layers and/or fully connected layers, $\mathbf{I}$ is the identity matrix and $\mathbf{W}_l \in \Re^{f_{in}\times\ f_{out}}$ where $f_{in} = W \times\ H \times\ C$ and $f_{out} = M$. $\|\cdot\|_F$ represents the Frobenius norm. In other words, equation 3 constraints orthogonality among filters in one layer, which makes the learned features have minimum correlation with each other, thus implicitly reduce the redundancy and enhance the diversity among the filters, especially those from the lower layers [32].

Besides, orthonormality constraints provide alternative solution other than $L2$ regularization to the exploration of weight space in learning process. It provides more probabilities by limiting set of parameters in an orthogonal space instead of inside a hypersphere.

3.3. Modulation

The dynamical isometry of signal propagation in neural networks has been mentioned and underlined several times [1, 30, 13], and it amounts to maintain the singular values of Jacobian, say $\mathbf{J} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$, to be around 1. In this section, we will analyze the variation of singular values of Jacobian through different types of layers in detail. We omit the layer index and bias term for simplicity and clarity.

For linear case, we have $\mathbf{y} = \mathbf{W}^T\mathbf{x}$, which shows that having dynamical isometry is equivalent to keep orthogonality since $\mathbf{J} = \mathbf{W}^T$ and $\mathbf{J}\mathbf{J}^T = \mathbf{W}^T\mathbf{W}$.

Next let us consider the activations after normalization transformation, $\mathbf{y} = \mathrm{BN}_{\gamma,\beta}(\mathbf{W}^T\mathbf{x})$, which we borrow the notation from [13]. Given the assumption that input dimension equals output dimension and both are d-dimension vectors, the Jacobian is

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{J}_{dd} \end{bmatrix}_{md\times md} \tag{4}$$

where each $\mathbf{J}_{kk}$ is a $m \times m$ square matrix, that is

$$\mathbf{J}_{kk} = \begin{bmatrix} \frac{\partial y_1^{(k)}}{\partial x_1^{(k)}} & \frac{\partial y_1^{(k)}}{\partial x_2^{(k)}} & \cdots & \frac{\partial y_1^{(k)}}{\partial x_m^{(k)}} \\ \frac{\partial y_2^{(k)}}{\partial x_1^{(k)}} & \frac{\partial y_2^{(k)}}{\partial x_2^{(k)}} & \cdots & \frac{\partial y_2^{(k)}}{\partial x_m^{(k)}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m^{(k)}}{\partial x_1^{(k)}} & \frac{\partial y_m^{(k)}}{\partial x_2^{(k)}} & \cdots & \frac{\partial y_m^{(k)}}{\partial x_m^{(k)}} \end{bmatrix} \tag{5}$$

Here $\frac{\partial y_i^{(k)}}{\partial x_j^{(k)}}$ denotes partial derivative of output of ith sample with respect to jth sample in kth component. The Jacobian of BN has its speciality that its partial derivatives are not only related with components of activations, but also with samples in one mini-batch. Because of each component k of activations is transformed independently by BN, $\mathbf{J}$ can be expressed with a blocked diagonal matrix as Eq. 4. Again since the independence among activations, we can analyse just one of d sub-Jacobians, *e.g.* $\mathbf{J}_{kk}$.

From equation 1 we can get the entries of $\mathbf{J}_{kk}$, which is

$$\frac{\partial y_j}{\partial x_i} = \rho\left[\Delta(i=j) - \frac{1+\hat{x}_i\hat{x}_j}{m}\right] \tag{6}$$

where $\rho = \frac{\gamma}{\sqrt{\sigma_B^2+\epsilon}}$ and $\Delta(\cdot)$ is the indicator operator. Here we still omit index k since dropping it brings no ambiguity.

Eq. 6 concludes obviously that $\mathbf{J}\mathbf{J}^T \neq \mathbf{I}$. So the orthonormality is not held after BN operation. Now the correlation among columns of $\mathbf{W}$ is directly impacted by normalized activations, while the corresponding weights determine these activations in turn, which results in a complicated situation. Fortunately, we can deduce the preferred equation according to subadditivity of matrix rank [3], which is

$$\mathbf{J} = \mathbf{P}^T\rho\begin{bmatrix} 1-\frac{\lambda_1}{m} & 0 & 0 & \cdots & 0 \\ 0 & 1-\frac{\lambda_2}{m} & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}_{m\times m}\mathbf{P} \tag{7}$$

where $\mathbf{P}$ is the matrix consists of eigenvectors of $\mathbf{J}$. λ_1 and λ_2 are two nonzero eigenvalues of $\mathbf{U}$, say $U_{ij} = 1 + \hat{x}_i\hat{x}_j,\ i = 1\cdots m,\ j = 1\cdots m$.

Eq. 7 shows us that $\mathbf{J}\mathbf{J}^T \approx \rho^2\mathbf{I}$ [5]. The approximation comes from first two diagonal entries in Eq. 7 which may

[5]The Jacobian after ReLU is amount to multiply a scalar with $\mathbf{J}$ [1], which we can merge it into ρ instead.

be close to zero. We think it is one of reasons that violate the perfect dynamic isometry and result in the degradation problem with this kind of non-full rank. Since value of ρ is determined by γ and σ_B, it is bounded as long as these two variables keep stable during the learning process, which achieves the so-called quasi-isometry [6].

Notice that ρ changes with γ and σ_B while γ and σ_B will change in every iteration. Based on the observation, we propose the scale factor ρ should be adjusted dynamically instead of fixing it like [1, 37, 30]. According to [30], when the nonlinearity is odd, so that the mean activity in each layer is approximately 0, neural population variance, or second order moment of output errors, can capture these dynamical properties quantitatively. ReLU nonlinearity is not satisfied but owing to the pseudo-normalization we can regard the errors propagated backwardly through BN as having zero mean, which makes the second order moment statistics reasonable.

4. Implementation Details

We insist to keep the orthonormality throughout the training process, so we implement this constraint both at initialization and in regularization. For a convolution parameter $\mathbf{W}_l \in \Re^{f_{in} \times f_{out}}$ of lth layer, we initialize subset of $\mathbf{W}$, say f_{in}-dimension vectors, on the first output channel. Then Gram-Schmidt process is applied to sequentially generate next orthogonal vectors channel by channel. Mathematically, generating n orthogonal vectors in d-dimension space which satisfies $n > d$ is ill-posed and, hence, impossible. So one solution is to avoid the fan-ins and fan-outs of kernels violating the principle, say $f_{in} \geq f_{out}$, in designing structures of networks; another candidate is group-wise orthogonalization proposed by us. If $f_{in} < f_{out}$, we divide the vectors into $\frac{f_{out}}{f_{in}} + f_{out} \mathrm{mod} f_{in}$ groups, orthogonalization is implemented within each group independently. We do not encourage the hybrid utilization of $L2$ regularization for those parameters of $f_{in} < f_{out}$ and orthonormal regularization for those of $f_{in} \geq f_{out}$. Forcing parameters to retract into inconsistent manifolds may cause convergence problems. Details can be referred in experiments.

For signal modulation, we compute the second order moment statistics of output errors between consecutive parametric layers (convolutional layer in our case) in each iteration. The scale factor ρ is defined as the square root of ratio of second order moment of higher layer, say q^{l+1}, to that, say q^l, of lower layer. However, if we modulate all the layers as long as $q^{l+1} \neq q^l$, then the magnitude of propagated signal will tend to be identical with the input error signal, which probably eliminate the variety encoded in the error signal. So we make a trade-off that the modulation only happens when the magnitudes of propagated signals of consecutive layers mismatch. Experiments show that it is a relatively reasonable and non-extreme modulation mechanism which has a capability of maintaining magnitude constancy for error signals.

5. Experiments

First of all, we must demonstrate that the core idea of this paper is to show that the proposed methods can be used to train extremely deep and plain CNNs and improve the performance drastically compared against prevalent stochastic gradient descent (SGD) with $L2$ regularization rather than achieving state-of-the-art performance in a certain dataset by all manner of means. Moreover, we try to show that the degradation problem of training a plain network reported in [11, 10, 35] can be partially solved by our methods.

5.1. Datasets and Protocols

Two representative datasets, CIFAR-10 [16] and ImageNet [27], are used in our experiments.

CIFAR-10. CIFAR-10 consists of $60,000$ 32×32 real world color images in 10 classes split into $50,000$ train and $10,000$ test images. All present experiments are trained on the training set and evaluated on the test set. Top-1 accuracy is evaluated.

ImageNet 2012 classification. For large-scale dataset, ImageNet 2012 classification dataset is used in our experiments. It consists of 1000 classes and there are 1.28 million training images and 50k validation images. Both top-1 and top-5 error rates are evaluated.

Protocol of CIFAR-10. To demonstrate that our proposed method can partially solve the degradation problem and show that the gap between deeper plain network and the shallower one can be shrunk or even removed, we aim to have fair comparison with the plain network in [11]. So we directly adopt their proposed architectures with minor modifications for both plain networks and residual networks. Specifically, the network inputs are 32×32 images, with the per-pixel mean subtracted and standard deviation divided. The first layer is 3×3 convolution and then following a stack of $6n$ 3×3 convolution layers, in which each convolution layer is accompanied by a BN layer and a ReLU layer (Fig. 1). While in the residual case, when size of feature maps doubles, *e.g.* 16 to 32, we use 3×3 projection shortcuts instead of identity ones. All the hyperparameters such as weight decay, momentum and learning rate are identical with [11]. Horizontal flip is the only data augmentation.

Protocol of ImageNet 2012 classification. The architectures in this protocol are also with a slight variation. Detailed architectures can be referred in Table 1. The hyperparameters are identical with those of CIFAR-10 protocol. 224×224 crops are randomly sampled on 256×256 images plus horizontal flip and color augmentation [17]. Mean of RGB is subtracted then scaling with a factor 0.017 (standard deviation of RGB). The mini-batch size is 256. Only the performances on validation set are reported.

layer name	output size	34-layer	101-layer
conv1	112×112	7×7, 64, stride 2	
		3×3 max pooling, stride 2	
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 1 \times 1, 64 \\ 3 \times 3, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 1 \times 1, 128 \\ 3 \times 3, 512 \end{bmatrix} \times 4$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 1 \times 1, 256 \\ 3 \times 3, 1024 \end{bmatrix} \times 23$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 1 \times 1, 512 \\ 3 \times 3, 2048 \end{bmatrix} \times 3$
	7×7	3×3, 1024	
	1×1	average pool, 1000-d fc, softmax	

Table 1. Architectures for ImageNet. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

5.2. Orthonormality Regularization Enhances the Magnitude of Signals

In this section, we design experiments to show that orthonormality can indeed enhance the magnitude of propagated signals in deep plain networks through decorrelating learned weights among different channels. A 44-layer plain network and CIFAR-10 dataset is adopted.

First we make statistics of average correlation among different channels over all the layers between two types of methods, say "msra" [12] initialization plus $L2$ regularization (abbr. as "msra+$L2$ reg") and our proposed orthonormal initialization and orthonormality regularization (abbr. as "ortho init+ortho reg"). Cosine distance $D_{cos}(\mathbf{x}, \mathbf{y})$ is considered to compute this value:

$$\bar{s} = \frac{1}{N} \sum_{l=1}^{D} \sum_{i=1}^{f_{out}} \sum_{j=i}^{f_{out}} D_{cos}(\mathbf{v}_i^{(l)}, \mathbf{v}_j^{(l)}) \tag{8}$$

where $\mathbf{v}_i^{(l)} \in \Re^{f_{in}}$ denotes ith kernel of $\mathbf{W}_l$ in lth layer and N is total computation count. From Fig. 2 we can see the variation of correlation among weights with iterations. Under the constraints of orthonormality, correlation of learned weights are forced into a consistent and relatively lower level (about 6×10^{-3}). On the contrary, "msra+$L2$ reg" cannot prevent from increasing correlation among weights as learning progresses. Finally, the correlation of "msra+$L2$ reg" is about 2.5 times higher than that of "ortho init+ortho reg", which demonstrates the effectiveness of orthonormality constraints.

Next we make statistics of variation of second order moments of back-propagated errors. Since the empirical risk will convergence as learning progresses, which results in smaller magnitude of loss value hence unscaled magnitude of error signals, we actually plot the ratio of second order moment of output error signals (input errors of first convolution layer) to that of input error signals (input errors of last convolution layer). Fig. 3 tells us that in first training phase (when learning rate is relatively large) the evolution of signal propagation is more insensitive than the in second and third training phases (when learning rate is small) because of mismatched order of magnitudes between learning rate and decay coefficient of regularizer (0.1 to 0.0001). How-

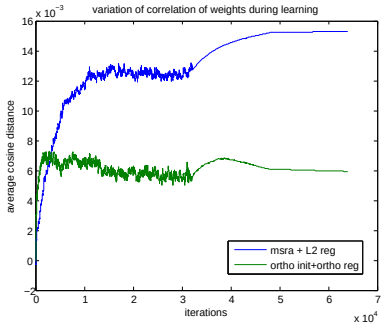

Figure 2. The variation of correlation among weight vectors in 44-layer plain network. The meaning of blue and green line can be refereed to the legend. One may be aware of at very first phase the correlation of blue line is lower than green one because of we allow negative correlations and "msra+$L2$ reg" method generates negative ones at first few iterations.

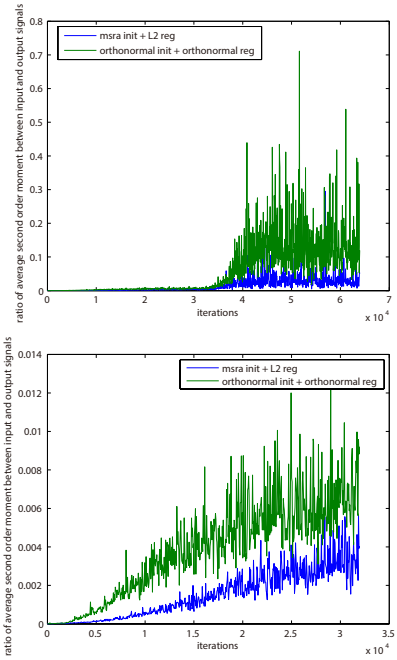

Figure 3. Variation of second order moments ratios of back-propagated errors in 44-layer plain network. Sample interval is 50 iterations for clearness. Orthonormality regularization can effectively alleviate the vanishing trend of magnitude of signals. **Top:** the plot over all iterations. **Bottom:** enlarged plot from 5th iteration to 32000th iteration (before learning rate is divided by 10).

ever, it shows the advantage of orthonormal regularization against $L2$ regularization no matter in which phase, especially in later phases. The magnitude of propagated signals is enhanced one order of magnitude by orthonormality. It is important to note that we omit the ratios of first five iterations in Fig. 3 since the disproportional order of magnitude. An interesting phenomenon is that all the magnitude of error signals is vanishing, *e.g*. ratio is less than 1, except for the initialization phase, in which the signals are amplified. We think randomness plays the key role for this phenomenon and it also provides evidence that makes us introduce orthonormality beyond initialization in optimizing extremely deep networks.

5.3. The Rationality of Modulation

In this section, we present our findings in training deep plain networks and aim to demonstrate modulation is a promising mechanism to train genuinely deep networks.

We find that a 44-layer network can be trained well just with orthonormality but a 110-layer one incurs seriously divergence, which states the accumulation effect mentioned in Sec. 3.1 by evidence. The proposed modulation is applied to train the 110-layer network and achieves distinct performance improvement against other one-order methods (see Table 2). The training methodology is a little tricky that we first apply with both orthonormality and modulation at the first n iterations, then the signals are regulated only through orthonormality until it converges. Keeping the magnitude of error signals to be isometric can easily be done by our modulation, but it is observed that this strategy undermines propagation of signals (81.6% *vs*. 73.5% on CIFAR-10). So when and how to modulation is an interesting and key research topic to totally solve the degradation problem.

In this paper the value of n is somewhat heuristic, which is derived from our observation to the evolution of ratios of second-order moment of output errors of each layer to the second-order moment of input errors at each iteration of training a 44-layer network. Fig. 4 reveals that it probably exists a potential evolution pattern in training deep networks. Actually we just shrink the degradation gap instead of eliminating it in training genuinely deep networks and one of our future work will focus on the methodology of modulation.

5.4. Results of Plain and Residual Network Architecture

To prove our proposed method has advantage against other methods integrated with the idea of adaptivity in training extremely deep plain networks, we compare it with six prevalent one-order methods in this section. We do not compare with second-order methods in consideration of implementation and memory practicality. Table 2 shows the performances[6]. We can see that most methods cannot handle relatively shallow networks well other than SGD and ours and all the methods except for ours cannot even converge in the deeper version. As pointed by [38], most one-order methods can only be a very effective method for optimizing certain types of deep learning architectures. So next we will focus on making comparison against more general SGD method. We also do not compare our method with other modulation methods, *e.g*. [1], because of they will fail convergence at the very first few iteration in such deep architecture

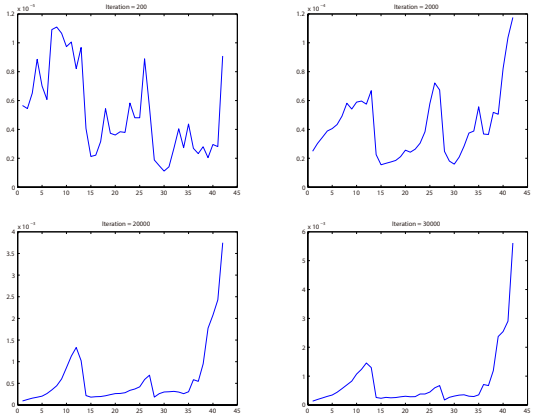

Figure 4. Evolution of backward signal propagation of a 44-layer plain network. X axis denotes layer index and Y axis denotes ratio of second-order moment of current layer to highest layer. We only present 200th, 2000th, 20000th and 30000th iteration, respectively. About after 2000 to 3000 iterations, the ratio trend to converge to a certain of stable evolution pattern shown in the 20000th and 30000th iterations.

Then we compare the performance with different regularizer in identical network architecture (ortho *vs*. $L2$ of a plain network), and further compare the performance of plain networks with residual networks have similar architectures (plain network with orthonormality *vs*. residual network with $L2$). Results are shown in Fig. 5. We can conclude that our proposed method has distinct advantage in optimizing plain networks and the orthonormality indeed can enhance magnitude of signal which alleviates gradient vanishing in training process.

To emphasize that orthonormality can be general to prevalent network architectures and large-scale datasets, we extend the experiments on ImageNet dataset. From Fig. 5 it shows the decreasing performance boost in ResNet-34 and almost comparative performance in ResNet-110. Compared with architectures on CIFAR-10, they have more channels, *e.g*. 64 *vs*. 2048, which introduces more redundan-

[6]We should mention that since the particularity of AdaDelta, which is less dependent on the learning rate, for more reasonable comparison, we ignore this hyper-parameter.

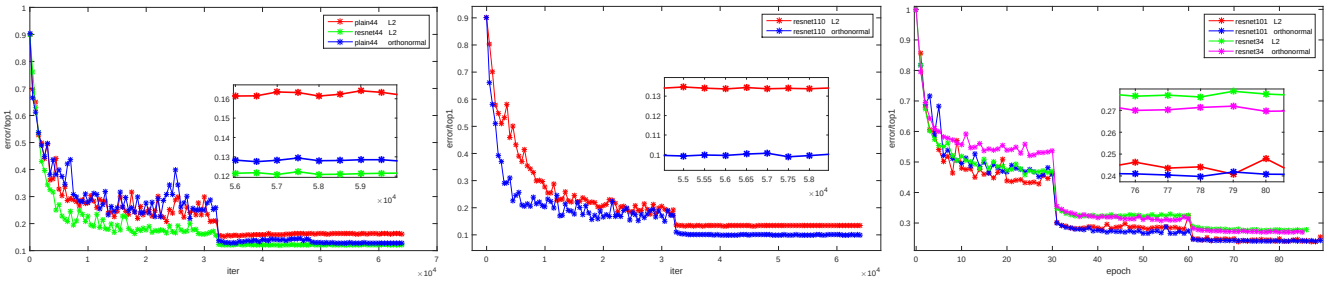

Figure 5. Miscellaneous performance comparisons about plain and residual networks on CIFAR-10 and ImageNet. **Left:** Performance comparisons of 44-layer plain network on CIFAR-10. One can see orthonormality boosts plain network to match the performance of residual architecture. **Middle:** Performance comparisons of 110-layer ResNet on CIFAR-10. Orthonormality helps convergence thus achieve higher performance. **Right:** Performance comparisons of 34-layer ResNet and 101-layer ResNet with different regularization on ImageNet.

Method	Top-1 Accuracy (%)		
	44-layer	110-layer	44-layer*
Nesterov[24]	85.0	10.18	61.9
AdaGrad[7]	77.86	30.3	36.1
AdaDelta[44]	70.56	66.48	52.6
Adam[14]	39.85	10.0	N/A
RmsProp[40]	10.0	10.0	N/A
SGD	84.14	11.83	65.2
Ours	**88.42**	**81.6**	**70.0**

Table 2. Performance comparison on CIFAR-10 and ImageNet of different optimization methods. Plain 44-layer and 110-layer networks are trained with these methods. All the common hyper-parameters are identical and specific ones are default (except for AdaDelta). $L2$ regularizer is applied for all the methods except for ours. N/A demonstrates the corresponding method cannot convergence at all and "*" means the methods are tested on ImageNet.

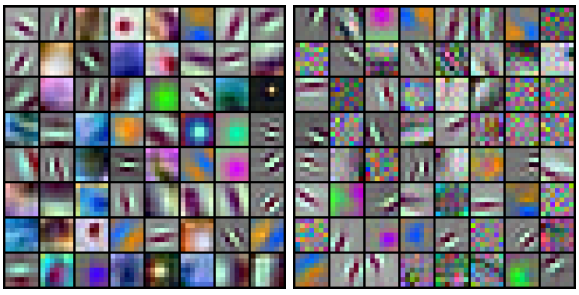

Figure 6. Weights visualization of first convolution layer in 34-layer residual network on ImageNet. **Left:** converged weights by $L2$ regularization. **Right:** converged weights by orthonormality.

cies among intra-layer's filter banks. Fig. 6 can be used to explain above results, so it probably be difficult for orthonormality to explore in parameter space with so many redundancies. The right sub-figure in Fig. 6 shows more noise-like feature maps than the left one, which inspires us to design thinner architectures in the future work.

6. Discussion and Conclusion

Recently we find that [21] has proposed similar ideas. They unify three types of kernel normalization methods into a geometric framework called kernel submanifolds, in which sphere, oblique and compact Stiefel manifolds (orthonormal kernels) are considered. The differences exists in three aspects: 1) The intrinsic explanations about the performance improvement is different, of which they mainly focus on regularization of models with data augmentation and learning of models endowed with geometric invariants; 2) The orthogonalization is different, of which they orthogonalize convolutional kernels within a channel while we do this among channel; 3) As the second statement tells, we believe that their proposed method still cannot handle the extremely deep plain networks. Besides, all the details and key steps to implement their methods are ambiguous that prevents from understanding and verifying it further.

Intrinsically, one can regard our proposed modulation as assigning each parametric layer an individual and adaptive learning rate. This kind of modulation can be more practical than local methods, *e.g.* second-order methods, while be more flexible than global ones, *e.g.* SGD. Besides, if we can approach some strategies to compensate the evanescent orthonomality as learning progresses, we believe that training a genuinely deep network will be available.

We propose a simple and direct method to train extremely deep plain networks with orthonormality and modulation. Furthermore, orthonormality reveals its generalization capability which can be applied in residual networks. Great performance boost is observed in experiments. However, the degradation problem is still not totally solved, which may be on condition understanding more comprehensively about the insights of signal modulation, reparametrization and novel constraints, *etc*. We hope our work will encourage more attentions on this problem.

References

[1] D. Arpit, Y. Zhou, B. U. Kota, and V. Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. *ICML*, 2016. 1, 2, 3, 4, 5, 7

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016. 2

[3] S. Banerjee and A. Roy. Linear algebra and matrix analysis for statistics. *Crc Press*, 2014. 4, 10

[4] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. 1

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 1

[6] D. J. Collins, R. I. Grigorchuk, P. F. Kurchanov, P. M. Cohn, and G. (Mathematik). *Combinatorial Group Theory, Applications to Geometry*. Springer, 1998. 5

[7] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 2011. 8

[8] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995. 2

[9] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, 9:249–256, 2010. 1, 2

[10] K. He and J. Sun. Convolutional neural networks at constrained time cost. *CVPR*, pages 5353–5360, 2015. 5

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015. 1, 2, 5

[12] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. pages 1026–1034, 2015. 1, 2, 6

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015. 1, 2, 3, 4

[14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference for Learning Representations*, 2015. 8

[15] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. *arXiv:1511.06856*, 2015. 1, 2, 3

[16] A. Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report*, 2009. 5

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25(2), 2012. 1, 5

[18] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv:1605.07648*. 1

[19] Y. Lecun, L. Bottou, G. B. Orr, and K. R. Mller. Efficient backprop. *Neural Networks Tricks of the Trade*, 1524(1):9–50, 2000. 2

[20] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 1

[21] O. Mete and O. Takayuki. Optimization on submanifolds of convolution kernels in cnns. *arXiv:1610.07008*, 2016. 8

[22] D. Mishkin and J. Matas. All you need is a good init. *Tetrahedron*, 69(14):3013–3018, 2015. 1, 2

[23] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. *ICML*, 2010. 1

[24] Y. Nesterov. A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/\sqrt{k})$. *Soviet Mathematics Doklady*, 1983. 8

[25] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. *arXiv preprint arXiv:1603.08695*, 2016. 1

[26] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2016. 1

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 1, 5

[28] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv:1602.07868*, 2016. 2, 3

[29] A. Saxe, J. McClelland, and S. Ganguli. Learning hierarchical category structure in deep neural networks. 2013. http://stanford.edu/~jlmcc/papers/SaxeMcCGanguli13CogSciProc.pdf. 2

[30] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120*, 2013. 1, 2, 3, 4, 5

[31] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *ICLR*, 2014. 1

[32] W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. *arXiv:1603.05201*, 2016. 4

[33] A. Shrivastava, A. Gupta, and R. B. Girshick. Training region-based object detectors with online hard example mining. *CoRR*, abs/1604.03540, 2016. 1

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 1, 2

[35] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv:1505.00387*, 2015. 1, 5

[36] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. *arXiv:1507.06228*, 2015. 1

[37] D. Sussillo and L. F. Abbott. Random walk initialization for training very deep feedforward networks. *arXiv:1412.6558*, 2014. 1, 2, 3, 5

[38] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. *Proceedings of the 30th International Conference on Machine Learning*, 2013. 7

[39] C. Szegedy, W. Liu, Y. Jia, and P. Sermanet. Going deeper with convolutions. *CVPR*, 2015. 1

[40] T. Tieleman and G. Hinton. Rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning. Technical report*, 2012. 8

[41] A. Veit, M. Wilber, and S. Belongie. Residual networks are exponential ensembles of relatively shallow networks. *arXiv:1605.06431*, 2016. 1

[42] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Craft objects from images. *arXiv preprint arXiv:1604.03239*, 2016. 1

[43] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv:1605.07146*. 1

[44] M. Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint*, 2012. 8

[45] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *ECCV*, 2014. 1

7. Quasi-isometry inference with Batch Normalization

For batch normalization (BN) layer, its Jacobian, denoted as $\mathbf{J}$, is not only related with components of activations (d components in total), but also with samples in one mini-batch (size of m).

Let $x_j^{(k)}$ and $y_i^{(k)}$ be kth component of jth input sample and ith output sample respectively and given the independence between different components, $\frac{\partial y_i^{(k)}}{\partial x_j^{(k)}}$ is one of $m^2 d$ nonzero entries of $\mathbf{J}$. In fact, $\mathbf{J}$ is a tensor but we can express it as a blocked matrix:

$$\mathbf{J} = \begin{bmatrix} \mathbf{D}_{11} & \mathbf{D}_{12} & \cdots & \mathbf{D}_{1m} \\ \mathbf{D}_{21} & \mathbf{D}_{22} & \cdots & \mathbf{D}_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{D}_{m1} & \mathbf{D}_{m2} & \cdots & \mathbf{D}_{mm} \end{bmatrix} \tag{9}$$

where each $\mathbf{D}_{ij}$ is a $d \times d$ diagonal matrix:

$$\mathbf{D}_{ij} = \begin{bmatrix} \frac{\partial y_i^{(1)}}{\partial x_j^{(1)}} & & & \\ & \frac{\partial y_i^{(2)}}{\partial x_j^{(2)}} & & \\ & & \ddots & \\ & & & \frac{\partial y_i^{(d)}}{\partial x_j^{(d)}} \end{bmatrix} \tag{10}$$

Since BN is a component-wise rather than sample-wise transformation, we prefer to analyse a variant of Eq. 9 instead of $\mathbf{D}_{ij}$. Note that by elementary matrix transformation, the m^2 $d \times d$ matrices can be converted into d $m \times m$ matrices:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{J}_{dd} \end{bmatrix} \tag{11}$$

and the entries of each $\mathbf{J}_{kk}$ is

$$\frac{\partial y_j}{\partial x_i} = \rho \left[\Delta(i=j) - \frac{1 + \hat{x}_i \hat{x}_j}{m} \right] \tag{12}$$

The notations of ρ, $\Delta(\cdot)$ and $\hat{x}_k$ have been explained in our main paper and here we omit the component index k for clarity. Base on the observation of Eq. 12, we separate the numerator of latter part and denote it as $U_{ij} = 1 + \hat{x}_i \hat{x}_j$.

Let $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, ..., \hat{x}_m)^T$, $\mathbf{e} = (1, 1, ..1)^T$, we have

$$\mathbf{U} = \mathbf{e}\mathbf{e}^T + \hat{\mathbf{x}}\hat{\mathbf{x}}^T \tag{13}$$

and

$$\mathbf{J}_{kk} = \rho(\mathbf{I} - \frac{1}{m}\mathbf{U}) \tag{14}$$

Recall that for any column vector $\mathbf{v}$, $\mathtt{rank}(\mathbf{v}\mathbf{v}^T) = 1$. According to the subadditivity of matrix rank [3], it implies that

$$\begin{aligned} \mathtt{rank}(\mathbf{U}) = \mathtt{rank}(\mathbf{e}\mathbf{e}^T + \hat{\mathbf{x}}\hat{\mathbf{x}}^T) \leq \\ \mathtt{rank}(\mathbf{e}\mathbf{e}^T) + \mathtt{rank}(\hat{\mathbf{x}}\hat{\mathbf{x}}^T) = 2 \end{aligned} \tag{15}$$

Eq. 15 tells us that $\mathbf{U}$ actually only has two nonzero eigenvalues, say λ_1 and λ_2, and we can formulate $\mathbf{U}$ as follow:

$$\mathbf{U} = \mathbf{P}^T \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & 0 & & \\ & & & \ddots & \\ & & & & 0 \end{bmatrix} \mathbf{P} \tag{16}$$

combined with Eq. 14, finally we get the equation of $\mathbf{J}_{kk}$ from the eigenvalue decomposition view, which is

$$\mathbf{J} = \mathbf{P}^T \rho \begin{bmatrix} 1 - \frac{\lambda_1}{m} & & & & \\ & 1 - \frac{\lambda_2}{m} & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \mathbf{P} \tag{17}$$

To show that $\mathbf{J}_{kk}$ probably is not full rank, we formulate the relationship between $\mathbf{U}^2$ and $\mathbf{U}$

$$\begin{aligned} \mathbf{U}^2 = (\mathbf{e}\mathbf{e}^T + \hat{\mathbf{x}}\hat{\mathbf{x}}^T)(\mathbf{e}\mathbf{e}^T + \hat{\mathbf{x}}\hat{\mathbf{x}}^T) = \mathbf{e}\mathbf{e}^T\mathbf{e}\mathbf{e}^T + \mathbf{e}\mathbf{e}^T\hat{\mathbf{x}}\hat{\mathbf{x}}^T \\ + \hat{\mathbf{x}}\hat{\mathbf{x}}^T\mathbf{e}\mathbf{e}^T + \hat{\mathbf{x}}\hat{\mathbf{x}}^T\hat{\mathbf{x}}\hat{\mathbf{x}}^T = m\mathbf{e}\mathbf{e}^T + (\sum_{i=1}^m \hat{x}_i)\mathbf{e}\hat{\mathbf{x}}^T \\ + (\sum_{i=1}^m \hat{x}_i)\hat{\mathbf{x}}\mathbf{e}^T + (\sum_{i=1}^m \hat{x}_i^2)\hat{\mathbf{x}}\hat{\mathbf{x}}^T \\ = m\mathbf{U} + (\sum_{i=1}^m \hat{x}_i)\mathbf{e}\hat{\mathbf{x}}^T + (\sum_{i=1}^m \hat{x}_i)\hat{\mathbf{x}}\mathbf{e}^T + (\sum_{i=1}^m \hat{x}_i^2 - m)\hat{\mathbf{x}}\hat{\mathbf{x}}^T \end{aligned} \tag{18}$$

Note that $\hat{x}_i \sim N(0,1)$, so we can regard the one-order and second-order accumulated items in Eq. 18 as approximately equaling the corresponding one-order and second-order statistical moments for relatively large mini-batch, from which we get $\mathbf{U}^2 \approx m\mathbf{U}$.

The relationship implies that $\lambda_1^2 \approx m\lambda_1$ and $\lambda_2^2 \approx m\lambda_2$. Since λ_1 and λ_2 cannot be zeros, it concludes that $\lambda_1 \approx \lambda_2 \approx m$ therefor $1 - \frac{\lambda_1}{m} \approx 0$ and $1 - \frac{\lambda_2}{m} \approx 0$ if batch size is sufficient in a statistical sense.