STACKING MORE LINEAR OPERATIONS WITH ORTHOGONAL REGULARIZATION TO LEARN BETTER

Weixiang Xu[1,2] *Jian Cheng*[1]

[1]NLPR, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences

ABSTRACT

How to improve the generalization of CNN models has been a long-lasting problem in the deep learning community. This paper presents a runtime parameter/FLOPs-free method to strengthen CNN models by stacking linear convolution operations during training. We show that overparameterization with appropriate regularization can lead to a smooth optimization landscape that improves the performance. Concretely, we propose to add a 1×1 convolutional layer before and after the original $k \times k$ convolutional layer respectively, without any non-linear activations between them. In addition, Quasi-Orthogonal Regularization is proposed to maintain the added 1×1 filters as orthogonal matrixes. After training, those two 1×1 layers can be fused into the original $k \times k$ layer without changing the original network architecture, leaving no extra computations at inference, i.e. parameter/FLOPs-free.

Index Terms— Over-parameterization, Model generalization, Orthogonal regularization

1. INTRODUCTION

A variety of strategies have been proposed to improve the generalization of CNN models. For example, data augmentations either with manual design [1] or automatical search [2] perform well on improving the generalization of different neural network architectures. Besides, Knowledge distillation [3, 4], which uses the prediction of large models to guide the training of small models, can effectively train DNNs to obtain higher accuracy. All these methods do not modify the original model architectures, and the improvements come from training procedure refinements. On the other hand, some works focus on designing plug-and-play modules that can be inserted into the well-designed architectures. For example, SE [5] and CBAM [6] enhance models' expressiveness with attention mechanisms. Though significant improvements in performance can be brought without redesigning network architectures, these methods introduce extra parameters/calculations not only during training but also at inference time, which defeats the purpose of deploying models on mobile devices.

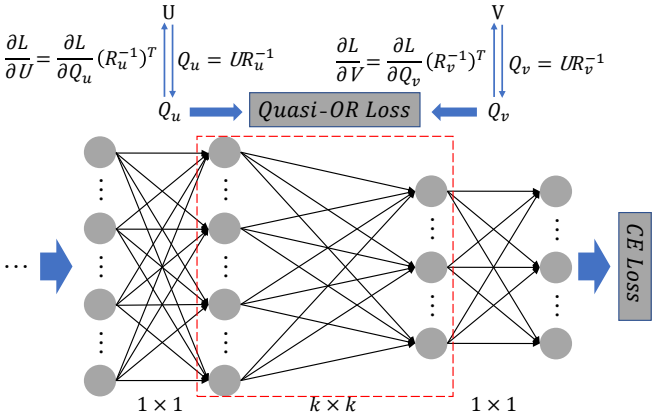

Fig. 1. Training procedure for Mergeable Linear Block with Quasi-Orthogonal Regularization. The part in the red dotted box denotes the original $k \times k$ convolution W. U and V are the added 1×1 filters. And it is the proxy tensors Q_U and Q_V that actually convolve with inputs during training. They are associated together through an upper triangular matrix R. After training, the orthogonal tensors Q_U and Q_V are merged into the original filters W, obtaining an equivalent filter $\widetilde{W}$.

In this work, we explore the advantages of overparameterization in CNNs by adding additional 1×1 filters with regularization before and after the original $k \times k$ filters. In addition, we enforce kernels of the additional 1×1 convolutional layers to be orthogonal, which can preserve energy and make spectra uniform during training by stabilizing the activation distribution. As a result, the model is promoted to explore its expressiveness, and higher performance can be obtained during training. In addition, we show that with a specific number of input/output channels, the added orthogonal 1×1 convolutions can be merged into the original convolutions after training, without changing the original architectures. **No extra parameters or computations are introduced during inference**.

Our motivation is inspired by the recent works on deep network optimization [7, 8]. [7] shows that two adjacent linear layers can provide implicit regularization for models to generalize well. And from an optimization view, [8] finds that

a faster convergence is obtained through replacing a matrix parameter with a product of matrices (without non-linearity between them). These investigations motivate us to explore the advantages of linear combinations in deep CNNs.

2. METHODOLOGY

A Standard CNN model can be seen as a composition of multiple nonlinear blocks. Each of these blocks consists of a linear transformation, followed by an element-wise non-linear activation $\sigma(\cdot)$, such as ReLU. We use $X_{j,:,:}$ to denote the j-th channel of input $X \in \mathbb{R}^{\text{channel}\times\text{height}\times\text{width}}$. Then the i-th channel of output Y can be calculated as the following linear transformation,

$$Y_{i,:,:} = \sum_{j=1}^{c} W^i_{j,:,:} \otimes X_{j,:,:} \tag{1}$$

Here, $\otimes$ denotes a 2D convolution operator. $W^i_{j,:,:}$ is the j-th channel of W^i. W^i is the i-th filter of 4D weights $W \in \mathbb{R}^{n\times c\times h\times w}$, where (n, c, h, w) are filter number, input channel number, kernel height and kernel width. Biases are omitted for simplicity.

2.1. Mergeable Linear Block

Just as we stated in our motivation, researchers [7, 8] find that two adjacent linear layers can provide implicit regularization for models to generalize well, with theoretical proof on a simple two-layer model. On this basis, we would like to explore the advantages of linear modules in modern architectures.

An intuitive way is to add extra convolutions to each nonlinear block of models. As shown in Figure 1, we insert one 1×1 convolutional layer before and after the original $k \times k$ convolutional layer, denoted by U and V respectively.

1×1 convolution, also known as pointwise convolution, has been widely explored for building new features through computing linear combinations of the input channels. It allows learnable interactions of cross channel information [9] and can complement tiny spatial details of feature maps [10]. As the formulation of convolution in Eq.1, 1×1 convolution between U and X can be written as,

$$\sum_{k=1}^{c_U} U^j_{k,:,:} \otimes X_{k,:,:} = \sum_{k=1}^{c_U} U^j_{k,:,:} \cdot X_{k,:,:} \tag{2}$$

In 1×1 convolution, $U^j_{k,:,:}$ is a real number rather than a 2D matrix. Therefore, the convolution operator $\otimes$ between $U^j_{k,:,:}$ and $X_{k,:,:}$ can be replaced by multiplication operator $\cdot$.

In order not to change the original model architecture after merging adjacent kernels, the shape of added convolution kernels is set to be 1×1. That is, $h_U = w_U = 1, h_V = w_V = 1$. As for filter number and input channel number of convolutions, we set $n_U = c_U = c_W$ and $n_V = c_V = n_W$. In this way, those two 1×1 layers can be fused into the original $k \times k$ layer after training to obtain a single $k \times k$ layer $\widetilde{W}$, without changing the original architecture. According to Eq.1 and Eq.2, inserting U before W can be formulated as below,

$$\begin{aligned} Y_{:,:,i} &= \sum_{j=1}^{c_W} W^i_{j,:,:} \otimes (\sum_{k=1}^{c_U} U^j_{k,:,:} \cdot X_{k,:,:}) \\ &= \sum_{j=1}^{c_W}\sum_{k=1}^{c_U} (W^i_{j,:,:} \cdot U^j_{k,:,:}) \otimes X_{k,:,:} = \sum_{k=1}^{c_U} \widetilde{W}^i_{k,:,:} \otimes X_{k,:,:} \end{aligned} \tag{3}$$

where $\widetilde{W} = \sum_{j=1}^{c_W} W^i_{j,:,:} \cdot U^j_{k,:,:}$ is the new equivalent filter, obtained by merging U and W. The last line of Eq.3 is the new equivalent convolution. Note that $\widetilde{W}$ has the same shape with the original W. The above derivation takes U and W as an example and it is the same for W and V.

2.2. Orthogonal Regularization

In the above section, we propose MLB. However, adding extra linear convolutions means increasing the depth of the model during training, which gives rise to the notorious problem of vanishing/exploding gradients [11, 12]. In this work, we alleviate the problem with the help of orthogonality regularization. We show that maintaining orthogonality of filters not only has norm-preserving property, but can also promote the model to explore its expressiveness with the help of stabilizing activation distribution and reducing redundancy in feature maps.

In this work, by introducing an appropriate proxy matrix, we propose a novel orthogonal regularization method that can trade off orthogonality against computation complexity. Specifically, given the added 1×1 convolutional kernel U, we perform QR decomposition on it to get an orthogonal matrix Q_U and an upper triangular matrix R_U. Several methods, such as Gram-Schmidt process or Givens rotations can be used for computing the QR decomposition:

$$U = Q_U R_U \tag{4}$$

As shown in Figure 1, it is proxy matrix Q_U rather than U that actually takes part in the forward and backward propagation, and its derivatives $\frac{\partial \ell}{\partial Q_U}$ can be obtained through back-propagation. Though U does not directly participate in inference, its derivatives can be obtained through chain rule:

$$\frac{\partial \ell}{\partial U} = \frac{\partial \ell}{\partial Q_U}\frac{\partial Q_U}{\partial U} = \frac{\partial \ell}{\partial Q_U} \cdot (R_U^{-1})^T \tag{5}$$

However, it is time-consuming to perform QR decomposition at every iteration. To alleviate it, we propose Quasi-Orthogonal Regularization (Quasi-OR). Concretely, according to Eq.4, orthogonal matrix can be calculated as

$$Q_U = U R_U^{-1} \tag{6}$$

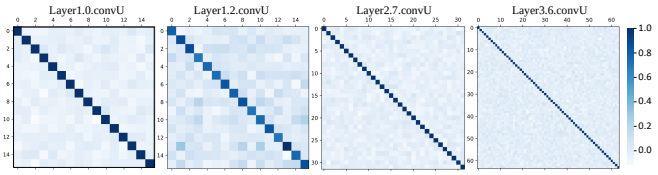

Fig. 2. Orthogonal property with Quasi-OR in some layers of ResNet-20 on CIFAR-100.

Here, U is updated by gradient descent with Eq.5. As for R_U, instead of updating it at each iteration, we introduce an update interval T. At i-th iteration, R_U^i is approximated as below,

$$R_U^{(i)} \approx R_U^{(kT)}, \text{ if } \quad kT < i < (k+1)T \tag{7}$$

where $k \in \{0, 1, 2...\}$. When $i = kT$, $R_U^{(kT)}$ is obtained through QR decomposition. In this way, computationally expensive QR is conducted only once within interval T. In this work, T is empirically set as 500 iterations.

The orthogonality loss within one nonlinear block (composed of U, W, V followed by a nonlinear function) is:

$$\mathcal{L}_{ortho} = \left\|Q_U^T Q_U - I\right\|_F^2 + \left\|Q_V^T Q_V - I\right\|_F^2 \tag{8}$$

3. EXPERIMENTS

In this section, we evaluate the proposed method in terms of qualitative and quantitative studies. Specific configurations are conducted for the added 1×1 convolutional layers: (1) we initialize them with the identity matrix, which is a special orthogonal matrix, keeping the spectrum of outputs the same with that of inputs. (2) The weight decay is set to 0.

3.1. Effectiveness Exploration

In this section, we explore the effect of the proposed method from a qualitative view. Specifically, we first show the effectiveness of Quasi-OR in making spectra uniform and stabilizing the activation distribution. Then we visualize the optimization landscape to further explore the advantages of MLB.

3.1.1. Quasi-OR

Figure 2 visualizes $Q^T Q$ after training to show how Quasi-OR can preserve orthogonality. Here, we train ResNet-20-MLB (equip the ResNet-20 with MLB by adding 1×1 kernels U,V before and after the original $k \times k$ convolutional layer respectively) on CIFAR-100 with Quasi-OR. The colors of pixels range from white to dark blue, indicating their values from 0 to 1. As shown in Figure 2, Quasi-OR behaves close to the goal $\left\|Q^T Q - I\right\|_F^2 = 0$.

We further show the effectiveness of Quasi-OR in making spectra uniform and stabilizing the activation distribution. The spectrum of a matrix is the set of its eigenvalues. In CNNs, given a convolutional kernel W with its transformation $Y = WX$, the spectrum of W measures how $\frac{\|Y\|}{\|X\|}$ varies with different X. We take ResNet-20-MLB on CIFAR-100 as an example. In order to show the relationship between spectrum and orthogonality, 1000 images are randomly selected from CIFAR-100 and used for forward and backward propagation. The results of $\frac{\|Y\|}{\|X\|}$ are shown in Figure 3, sorting from the largest to the smallest. Comparing with no regularization, our Quasi-OR makes the spectrum more uniform, guaranteeing that the energy of activations will not be amplified.

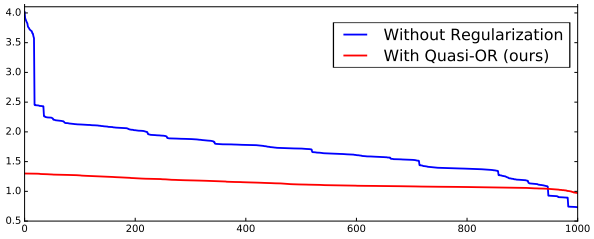

Fig. 3. The spectrum of weights. Taking ResNet-20 equipped with MLB as an example. $\frac{\|Y\|}{\|X\|}$ of the first Q_U is calculated and sorted from the largest to the smallest.

3.1.2. Optimization landscape

Following [13], we calculate the Lipschitzness and "effective" β-smoothness in training ResNet-20, which measures the stability and the smoothness of optimization landscape of loss function [14]. From the Figure 4, though gradient predictiveness of the original ResNet-20 (red shade) is thin enough compared with that without BatchNorm in [13], our method performs a thinner shade (green shade), indicating that the Lipschitzness of the loss function can be further improved. The above observation can also be explained by the reparametrization [13] of trainable parameters, which makes the underlying optimization more stable (measured by Lipschitzness). Concretely, from the original W to our $\widetilde{W} = \{U, W, V\}$, the new $\widetilde{W}$ is equivalent to reparameterizing W through two learnable orthogonal transformations.

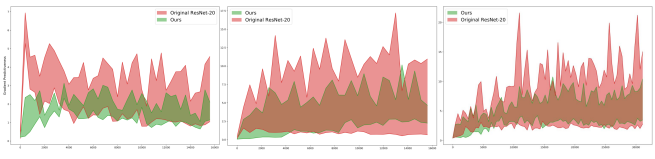

Fig. 4. The gradient predictiveness [13] of ResNet-20 with/without our method. The thinner shade shows the smoother loss landscape and thus less training difficulty.

Table 1. The Top-1 accuracy on CIFAR-10/100 across different architectures and the same architecture with different depth/width.

Model	CIFAR-10 Acc.(%)			CIFAR-100 Acc.(%)		
	Baseline	ACNet	Ours	Baseline	ACNet	Ours
VGG-8	92.88	–	+0.42	70.68	–	+0.70
VGG-16	93.58	+0.35	+0.96	74.16	+0.64	+0.91
ResNet-20	92.88	–	+0.60	69.63	–	+0.76
ResNet-56	94.17	+0.78	+0.89	73.44	+0.46	+1.08
WRN-16-2	94.31	–	+0.62	73.64	–	+0.52
DenseNet-40	93.0	+0.55	+0.77	72.45	+0.27	+0.86
DenseNet-100	94.23	–	+0.93	76.12	–	+1.21
SqueezeNet	92.63	–	+0.52	69.41	–	+0.54

Table 2. Performance on ImageNet across architectures.

Model	Top-1 Acc.(%)			Top-5 Acc.(%)		
	Baseline	Ours	$\Delta\uparrow$	Baseline	Ours	$\Delta\uparrow$
AlexNet	56.11	57.03	+0.92	80.2	80.83	+0.63
AlexNet-BN	60.54	61.07	+0.53	82.79	83.24	+0.45
ResNet-18	70.4	71.20	+0.8	89.45	90.01	+0.56
DenseNet-121	75.1	76.04	+0.94	92.29	92.87	+0.49

3.2. Results

We evaluate our method from a quantitative view by applying it to various architectures. Experiments are conducted on CIFAR-10/100 and the large-scale dataset ImageNet [15]. Standard PyTorch data augmentations and identical settings are performed for both baseline and ours for comparability. From the results in Table 1 and 2, we have the following observations: (1) Our methods can improve performance across different architectures, including the compact network SqueezeNet [16], indicating that our method can integrate well with different modern architectures. (2) Accuracy gains are consistent across different network depth/width with the same architecture, suggesting that the improvements induced by ours may be complementary to those obtained by simply increasing the depth of the base architecture. It is consistent with our explanation above that two learnable orthogonal transformations provide reparametrization of trainable parameters, making the underlying optimization more stable and smooth.

3.3. Ablation studies

In this section, we perform ablation studies to find out whether our proposed Quasi-OR is indeed specifically better for our MLB (and vice versa). In other words, we would like to show that MLB with Quasi-OR is not a simple combination like "A+B". Therefore, we conduct ablation experiments on whether Quasi-OR/MLB would also increase the performance of the original networks respectively. We take vanilla ResNet-56 on CIFAR-100 as our baseline and design three ablation experiments: 1) ResNet-56 + MLB, 2) ResNet-56 + Quasi-OR and 3) ResNet-56 + MLB + Quasi-OR. Training configurations are the same for them including random seed. From results in Table 3, we have the following findings,

Table 3. Ablation studies with ResNet-56 on CIFAR-100.

Method	Top-1(%)	$\Delta\uparrow$ (%)
ResNet-56(baseline)	73.58	–
ResNet-56 + MLB	73.62	+0.04
ResNet-56 + Quasi-OR	72.28	-1.34
ResNet-56 + MLB + Quasi-OR(ours)	**74.52**	**+0.94**

First, comparing the second line of Table 3 with the baseline, there is practically no improvement on accuracy. This phenomenon is inconsistent with the conclusion of [7, 8] who argue that implicit regularization can be provided by those adjacent linear layers for models to generalize well. This is because adding extra linear convolutions increases the depth of the model. As illustrated in Figure 3, when there is no regularization on those added 1×1 filters, it gives rise to the notorious problem of exploding gradients because of the large spectrum of those filters. The situation is particularly serious when three linear layers are blocked together in MLB without activations between them.

Second, in order to explore whether the accuracy improvement is simply attributed to the Quasi-Orthogonal Regularization, we design an ablation experiment by directly applying Quasi-OR on the original $k \times k$ filters. Comparing the third line of Table 3 with the baseline, directly applying Quasi-OR on $k \times k$ filters even hurts the accuracy. Previous works [17, 18] on the orthogonality of filters also notice the same phenomenon. Note that an essential precondition for the output channels of W to be orthogonal is that the shape of W should satisfy $n \geq chw$. In practical networks, the above precondition is often unsatisfied with some convolutional layers. For example, the shape of layer22$\sim$layer38 in ResNet-56 is $[32, 32, 3, 3]$ and the rank of $W^T W \in \mathbb{R}^{(chw)\times(chw)}$ is less than m so that $\left\|W^T W - I\right\| = 0$ will never be obtained for those layers. The unbalance between the objective and practical filter shapes may cause accuracy degeneration.

4. CONCLUSION

In this paper, we propose a simple yet effective training refinement method. By simply adding 1×1 convolutional layer with specific number of input/output channels before and after the $k \times k$ filters, they can be merged into the original filters to obtain an equivalent convolutional layer. Quasi-Orthogonal Regularization is proposed to further make spectra uniform and stabilize the activation distribution of added 1×1 convolutions. Experiments show that our approach demonstrates consistent improvements over various modern architectures on different datasets.

5. REFERENCES

[1] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[2] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le, "Autoaugment: Learning augmentation strategies from data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 113–123.

[3] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[4] Yonglong Tian, Dilip Krishnan, and Phillip Isola, "Contrastive representation distillation," *arXiv preprint arXiv:1910.10699*, 2019.

[5] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[6] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[7] Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro, "Implicit regularization in matrix factorization," in *Advances in Neural Information Processing Systems*, 2017, pp. 6151–6159.

[8] Sanjeev Arora, Nadav Cohen, and Elad Hazan, "On the optimization of deep networks: Implicit acceleration by overparameterization," *arXiv preprint arXiv:1802.06509*, 2018.

[9] Min Lin, Qiang Chen, and Shuicheng Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[10] Qiulin Zhang, Zhuqing Jiang, Qishuo Lu, Jia'nan Han, Zhengxin Zeng, Shang-hua Gao, and Aidong Men, "Split to be slim: An overlooked redundancy in vanilla convolution," *arXiv preprint arXiv:2006.12085*, 2020.

[11] Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal, "On orthogonality and learning recurrent networks with long term dependencies," *arXiv preprint arXiv:1702.00071*, 2017.

[12] Jianping Zhou, Minh N Do, and Jelena Kovacevic, "Special paraunitary matrices, cayley transform, and multidimensional orthogonal filter banks," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 511–519, 2006.

[13] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry, "How does batch normalization help optimization?," in *Advances in Neural Information Processing Systems*, 2018, pp. 2483–2493.

[14] Andrew M Saxe, James L McClelland, and Surya Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013.

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

[16] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[17] Di Xie, Jiang Xiong, and Shiliang Pu, "All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6176–6185.

[18] Nitin Bansal, Xiaohan Chen, and Zhangyang Wang, "Can we gain more from orthogonality regularizations in training deep networks?," in *Advances in Neural Information Processing Systems*, 2018, pp. 4261–4271.