Switchable Whitening for Deep Representation Learning

Supplementary Material

Xingang Pan[1], Xiaohang Zhan[1], Jianping Shi[2], Xiaoou Tang[1], and Ping Luo[1,3]

[1]CUHK-SenseTime Joint Lab, The Chinese University of Hong Kong

[2]SenseTime Group Limited [3]The University of Hong Kong

{px117, zx017, xtang, pluo}@ie.cuhk.edu.hk, shijianping@sensetime.com

Appendix

This document provides (1) back-propagation of ZCA whitening, (2) discussion for our network configurations, (3) the computational cost of SW, and (4) some style transfer visualization results.

A. Back-propagation for Whitening

In this section, we provide the back-propagation for Algorithm 2 of the paper.

Forward Pass. Let $\mathbf{X} \in \mathbb{R}^{C\times HW}$ be a sample of a mini-batch. Here the subscript n is omitted for clearance. Given the integrated mean $\hat{\boldsymbol{\mu}}$ and the integrated covariance $\hat{\boldsymbol{\Sigma}}$ in SW, the ZCA whitening is as follows:

$$\hat{\boldsymbol{\Sigma}} = \mathbf{D}\boldsymbol{\Lambda}\mathbf{D}^T \tag{1}$$

$$\mathbf{V} = \boldsymbol{\Lambda}^{-1/2}\mathbf{D}^T \tag{2}$$

$$\tilde{\mathbf{X}} = \mathbf{V}(\mathbf{X} - \hat{\boldsymbol{\mu}}\cdot\mathbf{1}^T) \tag{3}$$

$$\hat{\mathbf{X}} = \mathbf{D}\tilde{\mathbf{X}} \tag{4}$$

where $\mathbf{V}$ and $\tilde{\mathbf{X}}$ are intermediate variables for clarity in derivation.

Back-propagation. Based on the chain rule and the results in [1], $\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}}$ and $\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}}$ can be calculated as follows:

$$\frac{\partial L}{\partial \tilde{\mathbf{X}}} = \frac{\partial L}{\partial \hat{\mathbf{X}}}\mathbf{D} \tag{5}$$

$$\frac{\partial L}{\partial \mathbf{V}} = \frac{\partial L}{\partial \tilde{\mathbf{X}}}^T(\mathbf{X} - \hat{\boldsymbol{\mu}}\cdot\mathbf{1}^T)^T \tag{6}$$

$$\frac{\partial L}{\partial \boldsymbol{\Lambda}} = \frac{\partial L}{\partial \mathbf{V}}\mathbf{D}(-\frac{1}{2}\boldsymbol{\Lambda}^{-3/2}) \tag{7}$$

$$\frac{\partial L}{\partial \mathbf{D}} = \frac{\partial L}{\partial \mathbf{V}}\boldsymbol{\Lambda}^{-1/2}) + \frac{\partial L}{\partial \hat{\mathbf{X}}}\tilde{\mathbf{X}}^T \tag{8}$$

$$\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}} = \mathbf{D}\{[\mathbf{K}^T \odot (\mathbf{D}^T\frac{\partial L}{\partial \mathbf{D}})] + (\frac{\partial L}{\partial \boldsymbol{\Lambda}})_{diag}\}\mathbf{D}^T \tag{9}$$

$$\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}} = \frac{\partial L}{\partial \tilde{\mathbf{X}}}(-\mathbf{V}) \tag{10}$$

where L is the loss calculated via a loss function, $\mathbf{K} \in \mathbb{R}^{C\times C}$ is a 0-diagonal matrix with $\mathbf{K}_{ij} = \frac{1}{\sigma_i - \sigma_j}[i \neq j]$, and $\odot$ is element-wise matrix multiplication. The results are used to calculate the line 4, 5 of Algorithm 2 in the paper.

B. Discussion for Network Configurations

In our experiments, we replace part of the BN layers in ResNet to SW layers to save computation and to reduce redundancy. In this section we discuss the network configurations in detail.

CIFAR-10/100. For CIFAR, the ResNet has two convolution layers in a residual module. Thus we apply SW or other counterparts after the 1*st* and the {4n}*th* (n = 1,2,3,...) convolution layers. For example, in ResNet20, the normalization layers considered are the {1,4,8,12,16}*th* layers. We consider the 1*st* layer because [2] shows that it is effective to conduct whitening there. The last layer is not considered because it is a classifier where normalization is not needed.

ADE20K and Cityscapes. For semantic segmentation, the ResNet50 has a bottleneck architecture with a period of three layers, where the second layer provides a compact embedding of the input features. Therefore we apply SW after the second convolution layer of the bottleneck. Then the normalization layers considered are those at the 1*st* and the {3n}*th* (n = 1,2,3,...) layers. The residual blocks with 2048 channels are not considered to save computation, which also follows the rule in [3] that instance normalization should not be added in deep layers. Thus in ResNet50, the normalization layers considered are the {1,3,6,...,39}*th* layers, containing 14 layers in total.

ImageNet. And for ImageNet, the network configuration is similar to ResNet50 in semantic segmentation, except that we consider the 1*st* and the {6n}*th* (n = 1,2,3,...) layers

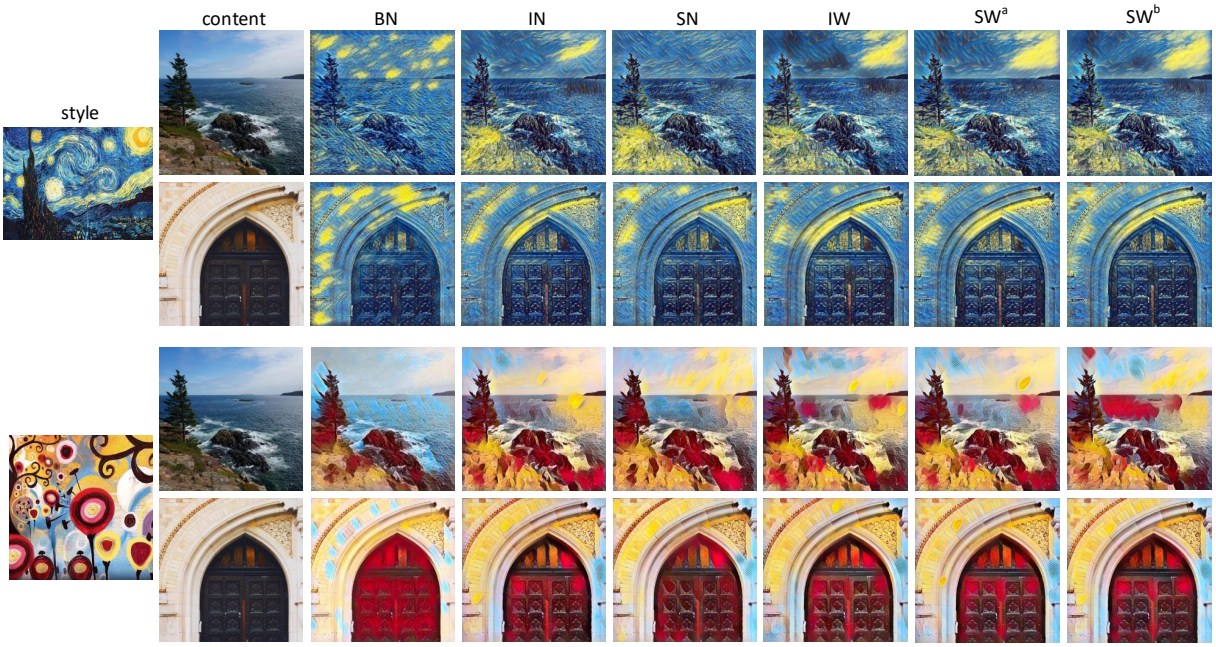

Figure 1. Visualization of style transfer using different normalization layers.

Table 1. Comparisons of computational complexity for ResNet50 on ImageNet. Input is an image with 224×224 size. 'F' denotes forward pass time (s) while 'F+B' denotes forward and backward pass time (s). The results are averaged over 100 iterations.

model	number of whitening	FLOPs ($\times 10^9$)	CPU[1]		GPU	
			F	F+B	F	F+B
ResNet50-BN	0	4.143	0.210	0.720	0.007	0.027
ResNet50-BW	7	4.185	0.259	0.793	0.353	0.513
ResNet50-SW^a	7	4.185	0.262	0.802	0.365	0.536
ResNet50-SW^b	7	4.188	0.268	0.817	0.379	0.577

Table 2. Comparisons of computational complexity for ResNet50 on Cityscapes. Input images have size 713×713, and the batchsize is 1 for CPU testing and 16 for multi-GPU testing. The results are averaged over 100 iterations.

model	number of whitening	FLOPs ($\times 10^9$)	CPU		GPU	
			F	F+B	F	F+B
ResNet50-BN	0	274.73	8.29	17.61	0.03	0.52
ResNet50-BW	7	275.30	8.42	17.72	0.87	1.09
ResNet50-SW^a	7	275.30	8.52	17.89	0.91	1.24
ResNet50-SW^b	7	275.34	8.70	18.48	1.27	1.64
ResNet50-BW	14	275.68	8.52	18.26	1.61	1.91
ResNet50-SW^a	14	275.68	8.80	18.49	1.72	2.21
ResNet50-SW^b	14	275.75	8.88	18.90	2.46	2.97

to further save computation. Thus the normalization layers considered are the $\{1,6,12,...,36\}th$ layers, containing 7 layers in total.

C. Computational Cost of SW

This section provides complexity analysis for SW. We compare SW with BN and BW with respect to FLOPs, CPU running time, and GPU running time. In our experiments, we adopt group whitening with group size $G = 16$ for SW and BW. Our implementation is based on Pytorch [4]. Results for ResNet50 on ImageNet and Cityscapes are shown in Table.1 and Table.2.

Computational Cost. For both datasets, SW brings about marginal FLOPs and CPU running time overhead compared with BN, as the main computational complexity lies in the convolution operation in the CNN.

For GPU testing, the running time of SW is comparable with BW. Nevertheless, our current implementation of SW on GPU is un-optimized, leading to larger running time than BN. For example, the GPU utilization is about 20%-40% for SW, and nearly 100% for BN. The bottleneck lies in singular value decomposition (SVD), and there is large room for faster implementation. However, optimizing GPU implementation for SW is beyond the scope of this work.

Using SW in Practice. In practice, we observe that the relative computational overhead on Cityscapes is smaller than that on ImageNet. This is because the current implementation of SW is sensitive to the number of training images, but not sensitive to image size. Therefore, in the current stage we recommend adopting SW in tasks that have large image

[1]The CPU is Intel Xeon CPU E5-2682 v4, and the GPU is NVIDIA Tesla V100.

size but less image number like semantic segmentation.

Moreover, one can trade off the model capacity and running time by tuning the number of SW layers. For example, ResNet50 with 14 SW layers achieves 39.8% and 76.2% mIoU on ADE20K and Cityscapes respectively, while ResNet50 with 7 SW layers has 39.6% and 75.7% mIoU on the two datasets. The ResNet50 with merely 7 SW layers still gives satisfactory results, and requires less GPU running time compared with ResNet50 with 14 SW layers, as shown in Table.2.

D. Style Transfer Results

Fig.1 provides visualization examples for image style transfer, where results of stylizing network with different normalization techniques are shown. It can be observed that the results of BN are worse than those of other methods, and SW produces comparably well stylizing images with IW. This shows that SW well adapts to the image style transfer task.

References

[1] C. Ionescu, O. Vantzos, and C. Sminchisescu. Training deep networks with structured layers by matrix backpropagation. *ICCV*, 2015. 1

[2] H. Lei, Y. Dawei, L. Bo, and D. Jia. Decorrelated batch normalization. *CVPR*, 2018. 1

[3] X. Pan, P. Luo, J. Shi, and X. Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. *ECCV*, 2018. 1

[4] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 2