Switchable Whitening for Deep Representation Learning

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

Abstract

Normalization methods are essential components in convolutional neural networks (CNNs). They either standardize or whiten data using statistics estimated in predefined sets of pixels. Unlike existing works that design normalization techniques for specific tasks, we propose Switchable Whitening (SW), which provides a general form unifying different whitening methods as well as standardization methods. SW learns to switch among these operations in an end-to-end manner. It has several advantages. First, SW adaptively selects appropriate whitening or standardization statistics for different tasks (see Fig.1), making it well suited for a wide range of tasks without manual design. Second, by integrating the benefits of different normalizers, SW shows consistent improvements over its counterparts in various challenging benchmarks. Third, SW serves as a useful tool for understanding the characteristics of whitening and standardization techniques.

We show that SW outperforms other alternatives on image classification (CIFAR-10/100, ImageNet), semantic segmentation (ADE20K, Cityscapes), domain adaptation (GTA5, Cityscapes), and image style transfer (COCO). For example, without bells and whistles, we achieve state-of-the-art performance with 45.33% mIoU on the ADE20K dataset. Code is available at https://github.com/XingangPan/Switchable-Whitening.

1. Introduction

Normalization methods have been widely used as a basic module in convolutional neural networks (CNNs). In various applications, different normalization techniques like Batch Normalization (BN) [13], Instance Normalization (IN) [31] and Layer Normalization (LN) [1] are proposed. These normalization techniques generally perform *standardization* that *centers* and *scales* features. Nevertheless, the features are not *decorrelated*, hence their correlation still exists.

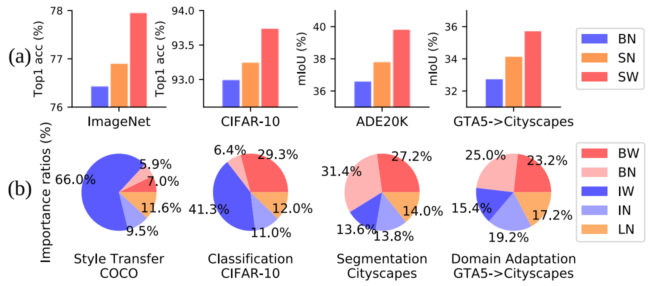


Figure 1. (a) SW outperforms its counterparts in a variety of benchmarks. (b) SW learns to select appropriate whitening or standardization methods in different tasks and datasets. The CNNs are ResNet50 for ImageNet and ADE20K, ResNet44 for CIFAR-10, and VGG16 for GTA5→Cityscapes. GTA5→Cityscapes indicates adapting from GTA5 to Cityscapes using domain adaptation.

Another type of normalization methods is *whitening*, which not only standardizes but also *decorrelates* features. For example, Decorrelated Batch Normalization (DBN) [17], or namely Batch Whitening (BW), whitens a mini-batch using its covariance matrix, which gives rise to better optimization efficiency than BN in image classification. Moreover, whitening features of an individual image is used in image style transfer [19] to filter out information of image appearance. Here we refer to this operation as instance whitening (IW). Despite their successes, existing works applied these whitening techniques separately to different tasks, preventing them from benefiting each other. Besides, whitening and standardization methods are typically employed in different layers of a CNN, which complicates model design.

To address the above issues, we propose *Switchable Whitening (SW)*. SW provides a general form that integrates different whitening techniques (*e.g.* BW, IW), as well as standardization techniques (*e.g.* BN, IN and LN). SW controls the ratio of each technique by learning their importance weights. It is able to select appropriate normalizers with respect to various vision tasks, as shown in Fig.1(b). For example, semantic segmentation prefers BW and BN, while IW is mainly chosen to address image diversity in image classification. Compared to semantic segmentation, domain adaptation selects more IW and IN, which alleviates domain discrepancy in CNN features. In image style transfer, IW dominates to handle image style variance.

SW can be inserted into advanced CNN architectures and effectively boosts their performances. Owing to the rich statistics and selectivity of SW, models trained with SW consistently outperform other counterparts in a number of popular benchmarks, such as CIFAR-10/100 [16] and ImageNet [5] for image classification, ADE20K [39] and Cityscapes [4] for semantic segmentation, domain adaptation between GTA5 [27] and Cityscapes, and image style transfer on COCO [22]. For example, when using ResNet50 [9] for ImageNet, ADE20K, and Cityscapes, as well as using VGG16 [29] for domain adaptation, SW significantly outperforms the BN-based baselines by 1.51%, 3.2%, 4.1%, and 3.0% respectively.

SW serves as a useful tool for analyzing the characteristics of these whitening or standardization techniques. This work answers two questions: (1) *Is IW beneficial for high-level vision tasks like classification and domain adaptation?* (2) *Is standardization still necessary when whitening is presented?* Our experiments suggest that (1) IW works extremely well for handling image appearance diversity and reducing domain gap, giving rise to better performance in high-level vision tasks; (2) Using BW+IW in SW performs comparably well compared to using all the normalizers mentioned above in SW, indicating that full whitening generally works well, and the requirement for standardization is marginal when whitening is presented.

Overall, our **contributions** are summarized as follows. (1) We propose Switchable Whitening (SW), which unifies existing whitening and standardization methods in a general form and learns to switch among them during training. (2) SW adapts to various tasks and is used as a new building block in advanced CNNs. We show that SW outperforms its counterparts in multiple challenging benchmarks. (3) SW could be used as a tool to analyze the effects and characteristics of different normalization methods, and the interactions between whitening and standardization. We will make the code of SW available and hope it would deepen our understanding on various normalization methods.

2. Related Work

Normalization. Existing normalization techniques generally performs *standardization*. For example, Batch Normalization (BN) [13] *centers* and *scales* activations using the mean and variance estimated over a mini-batch, accelerating training and enhancing generalization. In contrast, Instance Normalization (IN) [31] and Layer Normalization (LN) [1] standardize activations with statistics computed over each individual channel and all channels of a layer respectively. IN is mainly used in image generation [11, 31] while LN has been proved beneficial for training recurrent neural networks [1]. The above three normalizers are combined in Switchable Normalization (SN) [24] that learns the ratio of each one. The combination of BN and IN is also explored in IBN-Net [26] and Batch-Instance Normalization [25]. Besides, there have been other attempts to improve BN for small batch sizes such as Group Normalization [33], Batch Renormalization [12], and Batch Kalman Normalization [32]. All these normalization methods perform centering and scaling to the activations, whereas the correlation between activations remains, leading to sub-optimal optimization efficiency. Our work provides a general form that integrates both whitening and standardization techniques, having SN as a special case.

Whitening. Another paradigm towards improving optimization is *whitening*. Desjardins *et al.* [6] proposes Natural Neural Network, which implicitly whitens the activations to improve conditioning of the Fisher Information Matrix. This improves optimization efficiency of deep neural networks. Decorrelated Batch Normalization (DBN) [17] whitens features using covariance matrix computed over a mini-batch. It extends BN by decorrelating features. In this paper, we refer to DBN as Batch Whitening (BW) for consistency. Moreover, in the field of image style transfer, whitening and coloring operations are used to manipulate the image appearance [19, 28]. This is because the appearance of an individual image is well encoded in the covariance matrix of its features. We call whitening of an individual image as instance whitening (IW). In this work, we make the first attempt to apply IW in high-level vision tasks like image classification and semantic segmentation.

3. Switchable Whitening (SW)

We first present a general form of whitening as well as standardization operations, and then introduce SW.

3.1. A General Form

Our discussion is mainly based on CNNs, where the data have four dimensions. Let $\mathbf{X} \in \mathbb{R}^{C \times NHW}$ be the data matrix of a mini-batch, where N, C, H, W indicate the number of samples, number of channels, height, and width respectively. Here N, H and W are viewed as a single dimension for convenience. Let matrix $\mathbf{X}_n \in \mathbb{R}^{C \times HW}$ be the nth sample in the mini-batch, where $n \in \{1, 2, ..., N\}$. Then the whitening transformation $\phi : \mathbb{R}^{C \times HW} \to \mathbb{R}^{C \times HW}$ for a sample $\mathbf{X}_n$ could be formulated as

$$\phi(\mathbf{X}_n) = \mathbf{\Sigma}^{-1/2}(\mathbf{X}_n - \boldsymbol{\mu} \cdot \mathbf{1}^T) \tag{1}$$

where $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ are the mean vector and the covariance matrix calculated from the data, and $\mathbf{1}$ is a column vector of all ones. Note that different whitening methods could be achieved by calculating $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ using different sets of pixels. We discuss them in detail as below.

Batch Whitening (BW). In BW [17], the statistics are calculated in a mini-batch. Thus

$$\boldsymbol{\mu}_{bw} = \frac{1}{NHW}\mathbf{X} \cdot \mathbf{1}$$
$$\boldsymbol{\Sigma}_{bw} = \frac{1}{NHW}(\mathbf{X} - \boldsymbol{\mu} \cdot \mathbf{1}^T)(\mathbf{X} - \boldsymbol{\mu} \cdot \mathbf{1}^T)^T + \epsilon\mathbf{I} \quad (2)$$

where $\epsilon > 0$ is a small positive number to prevent a singular $\boldsymbol{\Sigma}_{bw}$. In this way, the whitening transformation ϕ whitens the data of the entire mini-batch, *i.e.*, $\phi(\mathbf{X})\phi(\mathbf{X})^T = \mathbf{I}$.

Instance Whitening (IW). In contrast, for IW [19], $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are calculated within each individual sample,

$$\boldsymbol{\mu}_{iw} = \frac{1}{HW}\mathbf{X}_n \cdot \mathbf{1}$$
$$\boldsymbol{\Sigma}_{iw} = \frac{1}{HW}(\mathbf{X}_n - \boldsymbol{\mu} \cdot \mathbf{1}^T)(\mathbf{X}_n - \boldsymbol{\mu} \cdot \mathbf{1}^T)^T + \epsilon\mathbf{I} \quad (3)$$

for *n* in $\{1, 2, ..., \mathrm{N}\}$. IW whitens each samples separately, *i.e.*, $\phi(\mathbf{X}_n)\phi(\mathbf{X}_n)^T = \mathbf{I}$.

Note that Eq.(1) also naturally incorporates standardization operations as its special cases. In the covariance matrix $\boldsymbol{\Sigma}$, the diagonal elements are the variance for each channel, while the off-diagonal elements are the correlation between channels. Therefore, by simply setting the off-diagonal elements to zeros, the left multiplication of $\boldsymbol{\Sigma}^{-1/2}$ equals to dividing the standard variance, so that Eq.(1) becomes standardization.

Batch Normalization (BN). BN[13] centers and scales data using the mean and standard deviation of a mini-batch. Hence its mean is the same as in BW *i.e.*, $\boldsymbol{\mu}_{bn} = \boldsymbol{\mu}_{bw}$. As discussed above, since BN does not decorrelate data, the covariance matrix becomes $\boldsymbol{\Sigma}_{bn} = diag(\boldsymbol{\Sigma}_{bw})$, which is a diagonal matrix that only preserves the diagonal of $\boldsymbol{\Sigma}_{bw}$.

Instance Normalization (IN). Similarly, in IN [31] we have $\boldsymbol{\mu}_{in} = \boldsymbol{\mu}_{iw}$ and $\boldsymbol{\Sigma}_{in} = diag(\boldsymbol{\Sigma}_{iw})$.

Layer Normalization (LN). LN [1] uses the mean and variance of all channels in a sample to normalize. Let μ_{ln} and σ_{ln} denote the mean and the variance, then $\boldsymbol{\mu}_{ln} = \mu_{ln}\mathbf{1}$ and $\boldsymbol{\Sigma}_{ln} = \sigma_{ln}\mathbf{I}$. In practice μ_{ln} and σ_{ln} could be calculated efficiently from $\boldsymbol{\mu}_{in}$ and $\boldsymbol{\Sigma}_{in}$ using the results in [24].

In Eq.(1), the inverse square root of the covariance matrix is typically calculated by using ZCA whitening,

$$\boldsymbol{\Sigma}^{-1/2} = \mathbf{D}\boldsymbol{\Lambda}^{-1/2}\mathbf{D}^T \quad (4)$$

where $\boldsymbol{\Lambda} = diag(\sigma_1, ..., \sigma_c)$ and $\mathbf{D} = [\boldsymbol{d}_1, ..., \boldsymbol{d}_c]$ are the eigenvalues and the eigenvectors of $\boldsymbol{\Sigma}$, *i.e.*, $\boldsymbol{\Sigma} = \mathbf{D}\boldsymbol{\Lambda}\mathbf{D}^T$, which is obtained via eigen decomposition.

So far we have formulated different whitening and normalization transforms in a general form. In the next section, we introduce switchable whitening based on this formulation.

3.2. Formulation of SW

For a data sample $\mathbf{X}_n$, a natural way to unify the aforementioned whitening and standardization transforms is to combine the mean and covariance statistics of those methods, and perform whitening using this unified statistics, giving rise to

$$SW(\mathbf{X}_n) = \hat{\boldsymbol{\Sigma}}^{-1/2}(\mathbf{X}_n - \hat{\boldsymbol{\mu}} \cdot \mathbf{1}^T) \quad (5)$$

$$\text{where} \quad \hat{\boldsymbol{\mu}} = \sum_{k\in\Omega} \omega_k \boldsymbol{\mu}_k, \quad \hat{\boldsymbol{\Sigma}} = \sum_{k\in\Omega} \omega'_k \boldsymbol{\Sigma}_k \quad (6)$$

Here Ω is a set of statistics estimated in different ways. In this work, we mainly focus on two cases, *i.e.*, $\Omega = \{\text{bw, iw}\}$ and $\Omega = \{\text{bw, iw, bn, in, ln}\}$, where the former switches between two whitening methods, while the later incorporates both whitening and standardization methods. ω_k are importance ratios to switch among different statistics. In practice, ω_k are generated by the corresponding control parameters λ_k via softmax function, *i.e.*, $\omega_k = \frac{e^{\lambda_k}}{\sum_{z\in\Omega} e^{\lambda_z}}$. And ω'_k are defined similarly using another group of control parameters λ'_k. This relieves the constraint of consistency between mean and covariance, which is a more general form.

Note that the above formulation incorporates SN [24] as its special case by letting $\Omega = \{\text{bn, in, ln}\}$. Our formulation is more flexible and general in that it takes into account the whole covariance matrix rather than only the diagonal. This provides the possibility of producing decorrelated features, giving rise to either better optimization conditioning or style invariance. SW could be easily extended to incorporate some other normalization methods like Batch Renormalization [12] or Group Normalization [33], which is out of the scope of this work.

3.3. Training and Inference

Switchable Whitening could be inserted extensively into a convolutional neural network (CNN). Let Θ be a set of parameters of a CNN, and Φ be a set of importance weights in SW. The importance weights are initialized uniformly, *e.g.* $\lambda_k = 1$. During training, Θ and Φ are optimized jointly by minimizing a loss function $\mathcal{L}(\Theta, \Phi)$ using back-propagation. The forward calculation of our proposed SW is presented in Algorithm 1 while the backward pass is presented in Appendix. For clearance, we use $\Omega = \{\text{bw, iw}\}$ as an illustrative example.

In the training phase, $\boldsymbol{\mu}_{bw}$ and $\boldsymbol{\Sigma}_{bw}$ are calculated within each mini-batch and used to update the running mean and running covariance as in Line 7 and 8 of Algorithm 1. During inference, the running mean and the running covariance are used as $\boldsymbol{\mu}_{bw}$ and $\boldsymbol{\Sigma}_{bw}$, while $\boldsymbol{\mu}_{iw}$ and $\boldsymbol{\Sigma}_{iw}$ are calculated independently for each sample.

Algorithm 1 Forward pass of SW for each iteration.

```
1: Input: mini-batch inputs $\mathbf{X} \in \mathbb{R}^{C\times NHW}$, where the $n$th sample in the batch is $\mathbf{X}_n \in \mathbb{R}^{C\times HW}$, $n \in \{1,2,...,N\}$; importance weights $\lambda_k$ and $\lambda'_k$, $k \in \{bw, iw\}$; expected mean $\boldsymbol{\mu}_E$ and expected covariance $\mathbf{\Sigma}_E$.
2: Hyperparameters: $\epsilon$, running average momentum $\alpha$.
3: Output: the whitened activations $\{\hat{\mathbf{X}}_n, n = 1,2,...,N\}$.
4: calculate: $\omega_{bw}, \omega_{iw} = Softmax(\lambda_{bw}, \lambda_{iw}), \omega'_{bw}, \omega'_{iw} = Softmax(\lambda'_{bw}, \lambda'_{iw})$.
5: calculate: $\boldsymbol{\mu}_{bw} = \frac{1}{NHW}\mathbf{X}\cdot\mathbf{1}$.
6: calculate: $\mathbf{\Sigma}_{bw} = \frac{1}{NHW}(\mathbf{X}-\boldsymbol{\mu}\cdot\mathbf{1}^T)(\mathbf{X}-\boldsymbol{\mu}\cdot\mathbf{1}^T)^T + \epsilon\mathbf{I}$.
7: update: $\boldsymbol{\mu}_E \leftarrow (1-\alpha)\boldsymbol{\mu}_E + \alpha\boldsymbol{\mu}_{bw}$.
8: update: $\mathbf{\Sigma}_E \leftarrow (1-\alpha)\mathbf{\Sigma}_E + \alpha\mathbf{\Sigma}_{bw}$.
9: for $n = 1$ to $N$ do
10:     calculate: $\boldsymbol{\mu}_{iw}^{(n)} = \frac{1}{HW}\mathbf{X}_n\cdot\mathbf{1}$.
11:     calculate: $\mathbf{\Sigma}_{iw}^{(n)} = \frac{1}{HW}(\mathbf{X}_n-\boldsymbol{\mu}\cdot\mathbf{1}^T)(\mathbf{X}_n-\boldsymbol{\mu}\cdot\mathbf{1}^T)^T + \epsilon\mathbf{I}$.
12:     calculate: $\hat{\boldsymbol{\mu}}_n = \sum_k \omega_k \boldsymbol{\mu}_k^{(n)}$, $\hat{\mathbf{\Sigma}}_n = \sum_k \omega'_k \mathbf{\Sigma}_k^{(n)}$, $k \in \{bw, iw\}$.
13:     execute eigenvalue decomposition: $\hat{\mathbf{\Sigma}}_n = \mathbf{D}\mathbf{\Lambda}\mathbf{D}^T$.
14:     calculate ZCA-whitening matrix: $\mathbf{U}_n = \mathbf{D}\mathbf{\Lambda}^{-1/2}\mathbf{D}^T$.
15:     calculate ZCA-whitened output: $\hat{\mathbf{X}}_n = \mathbf{U}_n(\mathbf{X}_n - \hat{\boldsymbol{\mu}}_n\cdot\mathbf{1}^T)$.
16: end for
```

In practice, the scale and shift operations are usually added right after the normalization or whitening transform to enhance the model's representation capacity. For SW, we follow this design to introduce scale and shift parameters γ and β as in BN.

3.4. Accelerates SW via Newton's Iteration

In practice, the GPU implementation of singular value decomposition (SVD) in current deep learning frameworks are inefficient, leading to much slower training and inference. To address this issue, we could resort to an alternative way to calculate $\hat{\mathbf{\Sigma}}^{-1/2}$, which is to use Newton's iteration, as in IterNorm [10]. Following [10], we normalize $\hat{\mathbf{\Sigma}}$ via $\hat{\mathbf{\Sigma}}_N = \hat{\mathbf{\Sigma}}/tr(\hat{\mathbf{\Sigma}})$. Then calculate $\hat{\mathbf{\Sigma}}_N^{-1/2}$ via the following iterations:

$$\begin{cases} \mathbf{P}_0 = \mathbf{I} \\ \mathbf{P}_k = \frac{1}{2}(3\mathbf{P}_{k-1} - \mathbf{P}_{k-1}^3\hat{\mathbf{\Sigma}}_N),\ k = 1,2,...,T \end{cases} \tag{7}$$

where T is the iteration number, and $\mathbf{P}_k$ will converge to $\hat{\mathbf{\Sigma}}_N^{-1/2}$. Finally, we have $\hat{\mathbf{\Sigma}}^{-1/2} = \hat{\mathbf{\Sigma}}_N^{-1/2}/\sqrt{tr(\hat{\mathbf{\Sigma}})}$. In this work, we set $T = 5$, which produces similar performance with the SVD version.

3.5. Analysis and Discussion

We have introduced the formulation and training of SW. Here we discuss some of its important properties and analyze its complexity.

Instance Whitening for Appearance Invariance. In style transfer, researchers have found that image appearance information (*i.e.* color, contrast, style *etc.*) is well encoded in the covariance matrix of features produced by CNNs [19]. In this work, we take the first attempt to induce appearance invariance by leveraging IW, which is beneficial for domain adaptation or high-level vision tasks like classification or semantic segmentation. Although IN also introduces invariance by standardizing each sample separately, the difference in correlation could be easily enlarged in highly non-linear deep neural networks. In IW, features of different samples are not only standardized but also whitened individually, giving rise to the same covariance matrix, *i.e.*, identity matrix. Therefore, IW has better invariance property than IN.

Switching between Whitening and Standardization. Our formulation of SW makes it possible to switch between whitening and standardization. For example, considering $\Omega = \{\text{bw, bn}\}$, *i.e.*, $\hat{\mathbf{\Sigma}} = \omega_{bw}\mathbf{\Sigma}_{bw} + \omega_{bn}\mathbf{\Sigma}_{bn}$, $(\omega_{bw} + \omega_{bn} = 1)$. As ω_{bn} grows larger, the diagonal of $\hat{\mathbf{\Sigma}}$ would remain the same, while the off-diagonal would be weaken. This would make the features less decorrelated after whitening. This is beneficial when the extent of whitening requires careful adjustment, which is an important issue of BW as pointed out in [17].

Group SW. Huang *et al.* [17] uses group whitening to reduce complexity and to address the inaccurate estimation of large covariance matrices. In SW we follow the same design, *i.e.*, the features are divided into groups along the channel dimension and SW is performed for each group. The importance weights λ_k could be shared or independent for each group. In this work we let groups of a layer share the same λ_k to simplify discussion.

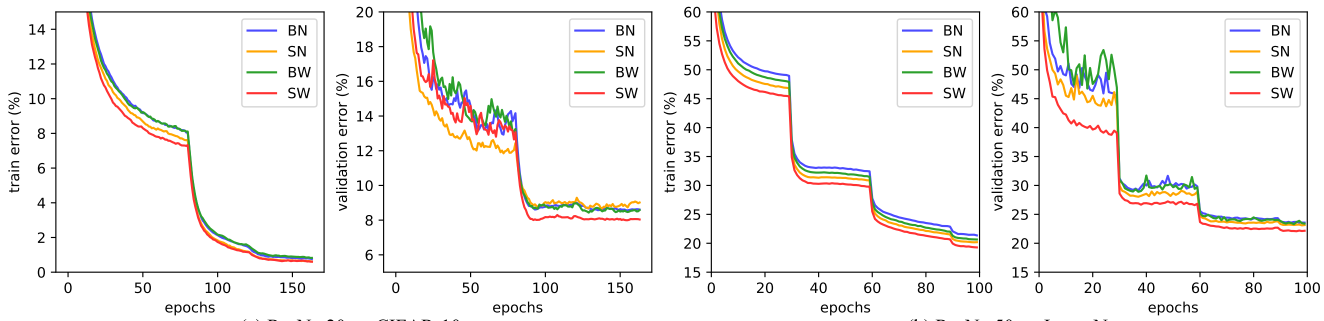
Table 1. Comparisons of computational complexity. N, C, H, W are the number of samples, number of channels, height, and width of the input tensor respectively. G denotes the number of channels for each group in group whitening.

Method	Computational complexity	
	w/o group	w/ group
BN,IN,LN,SN	$O(NCHW)$	$O(NCHW)$
BW	$O(C^2\text{max}(NHW, C))$	$O(CG\text{max}(NHW, G))$
IW	$O(NC^2\text{max}(HW, C))$	$O(NCG\text{max}(HW, G))$
SW	$O(NC^2\text{max}(HW, C))$	$O(NCG\text{max}(HW, G))$

Complexity Analysis. The computational complexities for different normalization methods are compared in Table 1. The flop of SW is comparable with IW. And applying group whitening could reduce the computation by C/G times. Usually we have $HW > G$, thus the computation cost of SW and BW would be roughly the same (*i.e.*, $O(CGNHW)$).

4. Experiments

We evaluate SW on image classification (CIFAR-10/100, ImageNet), semantic segmentation (ADE20K, Cityscapes), domain adaptation (GTA5, Cityscapes), and image style transfer (COCO). For each task, SW is compared with previous normalization methods. We also provide results of instance segmentation in the Appendix.



(a) ResNet20 on CIFAR-10 (b) ResNet50 on ImageNet

Figure 2. Training and validation error curve on CIFAR-10 and ImageNet. Models with different normalization methods are reported. Here SW has $\Omega = \{\text{bw, iw}\}$.

Table 2. Test errors (%) on CIFAR-10/100 and ImageNet validation sets [16]. For each model, we evaluate different normalization or whitening methods. SW^a and SW^b correspond to $\Omega = \{\text{bw, iw}\}$ and $\Omega = \{\text{bw, iw, bn, in, ln}\}$ respectively. Results on CIFAR are averaged over 5 runs.

Dataset	Method	BN	SN	BW	SW^a	SW^b
CIFAR-10	ResNet20	8.45	8.34	8.28	**7.64**	7.75
	ResNet44	7.01	6.75	6.83	**6.27**	6.35
	ResNet56	6.88	6.57	6.62	**6.07**	6.25
	ResNet110	6.21	5.97	5.99	**5.69**	5.78
CIFAR-100	ResNet20	32.09	32.28	32.44	31.00	**30.87**
	ResNet110	27.32	27.25	27.76	26.64	**26.48**
ImageNet	ResNet50 (top1)	23.58	23.10	23.31	22.10	**22.07**
	ResNet50 (top5)	7.00	6.55	6.72	5.96	**5.91**

4.1. Classification

CIFAR-10, CIFAR-100 [16] and ImageNet [5] are standard image classification benchmarks. Our training policies and settings are the same as in [9].

Implementation. We evaluate different normalization methods based on standard ResNet [9]. Note that introducing whitening after all convolution layers of ResNet is redundant and would incur a high computational cost, as also pointed out in [17]. Hence we replace part of the BN layers in ResNet to the desired normalization layers. For CIFAR, we apply SW or other counterparts after the 1*st* and the $\{4\text{n}\}$*th* (n = 1,2,3,...) convolution layers. And for ImageNet, the normalization layers considered are those at the 1*st* and the $\{6\text{n}\}$*th* (n = 1,2,3,...) layers. The residual blocks with 2048 channels are not considered to save computation. More discussions for such choices could be found in supplementary material.

The normalization layers studied here are BN, SN, BW, and SW. For SW, we consider two cases: $\Omega = \{\text{bw, iw}\}$ and $\Omega = \{\text{bw, iw, bn, in, ln}\}$, which are denoted as SW^a and SW^b respectively. In all experiments, we adopt group whitening with group size $G = 16$ for SW and BW. Since [23] shows that applying early stop to the training of SN reduces overfitting, we stop the training of SN and SW at the 80*th* epoch for CIFAR and the 30*th* epoch for ImageNet.

Results. The results are given in Table. 2 and the training curves are shown in Fig. 2. In both datasets, SW^a and SW^b show better results and faster convergence than BN, SN, and BW over various network depth. Specifically, with only 7 SW^b layers, the top1 and top5 error of ResNet50 on ImageNet is significantly reduced by 1.51% and 1.09%. This performance is comparable with the original ResNet152 which has 5.94% top5 error.

Our results reveal that combining different normalization methods in a suitable manner surpasses every single normalizer. For example, the superiority of SW^b over SN attributes to the better optimization conditioning brought out by whitening. And the better performance of SW^a over BW shows that instance whitening is beneficial as it introduces style invariance. Moreover, SW^a and SW^b perform comparably well, which indicates that full whitening generally performs well, and the need for standardization is marginal while whitening is presented.

Discussions. SW has two groups of importance weights λ_k and λ'_k. We observe that allowing λ_k and λ'_k to share weight produces slightly worse results. For example, ResNet20 has 8.17% test error when using SW with shared importance weights. We conjecture that mean and covariance have different impacts in training, and recommend to maintain independent importance weights for mean and covariance.

Note that IW is not reported here because it generally produces worse results due to diminished feature discrimination. For example, ResNet20 with IW gives 12.57% test error on CIFAR-10, which is worse than other normalization methods. This also implies that SW borrows the benefits of different normalizers so that it could outperform any individual of them.

4.2. Semantic Segmentation

We further verify the scalability of our method on ADE20K [39] and Cityscapes [4], which are standard and challenging semantic segmentation benchmarks. We evaluate SW based on ResNet and PSPNet [37].

Table 3. Results on Cityscapes and ADE20K datasets. 'ss' and 'ms' indicate single-scale and multi-scale test respectively.

Method	ADE20K		Cityscapes	
	mIoU_{ss}	mIoU_{ms}	mIoU_{ss}	mIoU_{ms}
ResNet50-BN	36.6	37.9	72.1	73.4
ResNet50-SN	37.8	38.8	75.0	76.2
ResNet50-BW	35.9	37.8	72.5	73.7
ResNet50-SW^a	**39.8**	**40.8**	**76.2**	**77.1**
ResNet50-SW^b	**39.8**	40.7	76.0	77.0

Table 4. Comparison with advanced methods on the ADE20K validation set. * indicates our implementation.

Method	mIoU(%)	Pixel Acc.(%)
DilatedNet [35]	32.31	73.55
CascadeNet [40]	34.90	74.52
RefineNet [21]	40.70	-
PSPNet101 [37]	43.29	81.39
SDDPN [20]	43.68	81.13
WiderNet [34]	43.73	81.17
PSANet101 [38]	43.77	81.51
EncNet [36]	44.65	81.69
PSPNet101*	43.59	81.41
PSPNet101-SW^a	**45.33**	**82.05**

Implementation. We adopt the same ResNet architecture, training setting, and data augmentation scheme as in [37]. The normalization layers considered are the 1*st* and the {3n}*th* (n = 1,2,3,...) layers except those with 2048 channels, resulting in 14 normalization layers for ResNet50. Since overfitting is not observed in these two benchmarks, early stop is not used here. The BN and BW involved are synchronized across multiple GPUs.

Results. Table.3 reports mIoU on the validation sets of the two benchmarks. For ResNet50, simply replacing part of BN with SW would significantly boost mIoU_{ss} by 3.2% and 4.1% for ADE20K and Cityscapes respectively. SW also notably outperforms SN and BW, which is consistent with the results of classification.

Furthermore, we show that SW could improve even the most advanced models for semantic segmentation. We apply SW to PSPNet101 [37], and compare with other methods on the ADE20K dataset. The results are shown in Table.4. Simply using some SW layers could improve the strong PSPNet by 1.74% on mIoU. And our final score, 45.33%, outperforms other more advanced semantic segmentation methods like PSANet [38] and EncNet [36].

Computational cost. While the above implementation of SW is based on SVD, they can be accelerated via Newton's iteration, as discussed in section 3.4. As shown in Table.5, the GPU running time is significantly reduced when using iterative whitening, while the performance is comparable to the SVD version. Note that in this Table, the ResNet-50-SW^a in Cityscapes has the same configuration as in ImageNet, *i.e.*, has 7 SW layers. Compared with the 14 layer version, this further saves computation cost, while

Table 5. Performance and running time of ResNet50 with different normalization layers on ImageNet and Cityscapes datasets. We report the GPU running time per iteration during training. The GPU we use is NVIDIA Tesla V100.

Method	Whitening	ImageNet		Cityscapes	
		error(%)	time(s)	mIoU(%)	time(s)
ResNet50-BN	-	23.58	0.27	72.1	0.52
ResNet50-BW	svd	23.31	0.79	72.4	1.09
ResNet50-SW^a	svd	22.10	1.04	75.7	1.24
ResNet50-SW^a	iterative	22.07	0.36	76.0	0.67

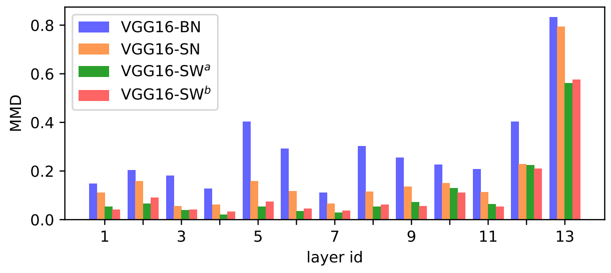


Figure 3. MMD distance between Cityscapes and GTA5.

still achieves satisfactory results.

4.3. Domain Adaptation

The adaptive style invariance of SW making it suitable for handling appearance discrepancy between two image domains. To verify this, we evaluate SW on domain adaptation task. The datasets employed are the widely used GTA5 [27] and Cityscapes [4] datasets. GTA5 is a street view dataset generated semi-automatically from the computer game Grand Theft Auto V (GTA5), while Cityscapes contains traffic scene images collected from the real world.

Implementation. We conduct our experiments based on the AdaptSegNet [30] framework, which is a recent state-of-the-art domain adaptation approach. It adopts adversarial learning to shorten the discrepancy between two domains with a discriminator. The segmentation network is DeepLab-v2 [3] model with VGG16 [29] backbone. The training setting is the same as in [30].

Note that the VGG16 model has five convolutional groups, where the number of convolution layers for these groups are {2,2,3,3,3}. We add SW or its counterparts after the first convolution layer of each group, and report the results using different normalization layers.

Results. Table.6 reports the results of adapting GTA5 to Cityscapes. The models with SW achieve higher performance when evaluated on a different image domain. Particularly, compared with BN, and SN, SW^a improves the mIoU by 3.0%, and 1.6% respectively.

To understand how SW performs better under cross-domain evaluation, we analyze the maximum mean discrepancy (MMD)[7] of deep features between the two datasets. MMD is a commonly used metric for evaluating domain discrepancy. Specifically, we use the MMD with Gaussian

Table 6. Results of adapting GTA5 to Cityscapes. mIoU of models with different normalization layers are reported.

Method	road	sidewalk	building	wall	fence	pole	light	sign	veg	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
AdaptSetNet-BN	88.3	42.7	74.9	22.0	14.0	16.5	17.8	4.2	83.5	34.3	72.1	44.8	1.7	76.9	18.0	**6.7**	0.0	3.0	0.1	32.7
AdaptSetNet-SN	87.0	41.6	77.5	21.2	**20.0**	**18.3**	20.9	**8.3**	82.4	35.4	72.6	48.4	1.4	81.1	**18.7**	5.2	0.0	8.4	0.0	34.1
AdaptSetNet-SW^a	**91.8**	50.2	78.1	**25.3**	17.5	17.5	**21.4**	6.2	83.4	36.6	74.0	**50.7**	**7.4**	83.4	16.7	6.3	0.0	**10.4**	**0.8**	**35.7**
AdaptSetNet-SW^b	**91.8**	**50.5**	**78.4**	23.5	16.5	17.2	19.8	5.5	**83.6**	**38.4**	**74.6**	48.9	5.3	**83.6**	17.6	3.9	**0.1**	7.7	0.7	35.1

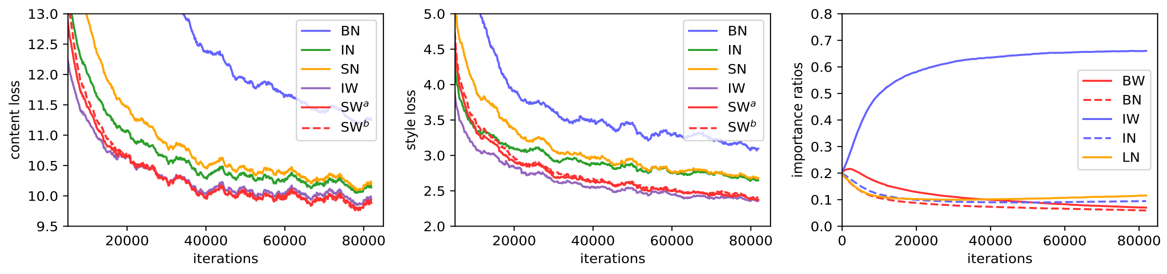


Figure 4. Training loss in style transfer and the learned importance ratios of SW^b. The importance ratios are averaged over all SW^b layers in the image stylizing network.

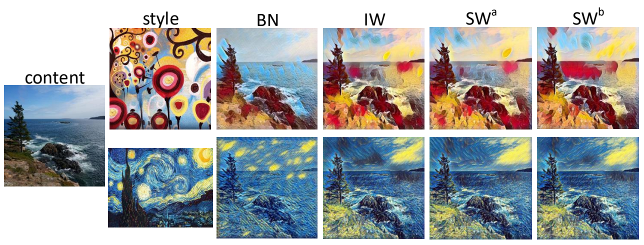


Figure 5. Visualization of style transfer using different normalization layers.

kernels as in [18]. We calculate the MMD for features of the first 13 layers in VGG16 with different normalization layers. The results are shown in Fig.3. Compared with BN and SN, SW significantly reduces MMD for both shallow and deep features. This shows that the IW introduced effectively reduces domain discrepancy in the CNN features, making the model easier to generalize to other data domains.

4.4. Image Style Transfer

Thanks to the rich statistics, SW could work not only in high-level vision tasks, but also in low-level vision tasks like image style transfer. To show this, we employ a popular style transfer algorithm [15]. It has an image stylizing network trained with the content loss and style loss calculated by a loss network. The MS-COCO dataset [22] is used as content images while the style images selected are *candy* and *starry night*. We follow the same training policy as in [15], and adopt different normalization layers for the image stylizing network.

Results. The training loss curve is shown in Fig.4. As revealed in former works, IW and IN perform better than BN. Besides, we observe that IW has smaller content loss and style loss than IN, which verifies that IW works better in manipulating image style. Although SW converges slower than IW at the beginning, it soon catches up with IW as SW learns to select IW as the normalizer. Moreover, SW has smaller content loss than IW when the training converges, as BW preserves important content information.

Qualitative examples of style transfer using different normalization layers are shown in Fig.5. BN produces poor stylization images, while IW gives satisfactory results. SW works comparably well with IW, showing that SW is able to select appropriate normalizer according to the task. More examples are provided in supplementary material.

4.5. Analysis on SW

In order to understand the behavior of SW, in this section we study its learning dynamics and the learned importance ratios.

Learning Dynamics. The importance ratios of SW is initialized to have uniform values, i.e. 0.5 for $\Omega = \{\text{bw, iw}\}$ and 0.2 for $\Omega = \{\text{bw, iw, bn, in, ln}\}$. To see how the ratios of SW in different layers change during training, we plot the learning curves of ω_k and ω'_k in Fig.6 and Fig.7. It can be seen that the importance ratios shift quickly at the beginning and gradually become stable. There are several interesting observations. (1) The learning dynamics vary across different tasks. In CIFAR-10, SW mostly selects IW and occasionally selects BW, while in Cityscapes BW or BN is mostly chosen. (2) The learning behaviours of SW across different layers tend to be distinct rather than homogeneous. For example, in Fig.7 (a), SW selects IW for layer {15, 21, 39}, and BW for the rest except for layer {6, 9} where the ratios keep uniform. (3) The behaviors of ω_k and ω'_k are mostly coherent and sometimes divergent. For instance, in layer {15, 21} of Fig.7, ω_k chooses IW while ω'_k chooses BW or BN. This implies that $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are not necessarily have to be consistent, as they might have different impacts

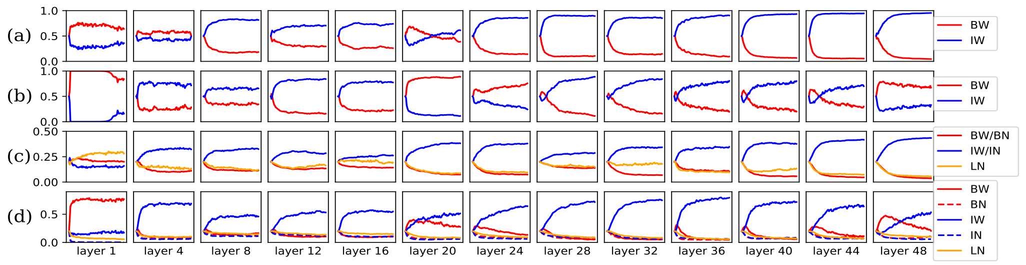


Figure 6. Learning curve of importance weights in ResNet56 on CIFAR-10. (a) and (b) show ω_k and ω'_k in SW with $\Omega = \{\text{bw}, \text{iw}\}$. (c) and (d) correspond to ω_k and ω'_k in SW with $\Omega = \{\text{bw}, \text{iw}, \text{bn}, \text{in}, \text{ln}\}$.

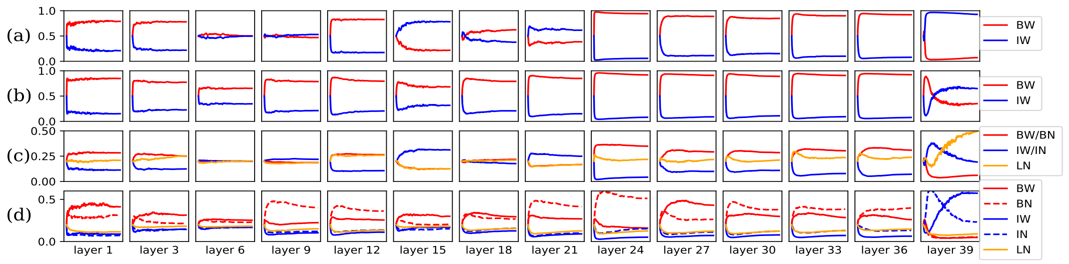


Figure 7. Learning curve of importance weights in ResNet50 on Cityscapes. (a)(b)(c)(d) have the same meanings as in Fig.6.

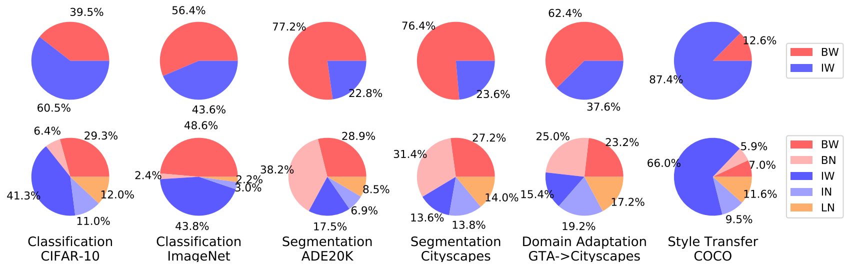


Figure 8. Learned importance ratios of SW in various tasks. Above and below correspond to $\Omega = \{\text{bw}, \text{iw}\}$ and $\Omega = \{\text{bw}, \text{iw}, \text{bn}, \text{in}, \text{ln}\}$ respectively.

in training.

Importance Ratios for Various Tasks.

We further analyze the learned importance ratios in SW for various tasks, as shown in Fig.8. The results are obtained by taking average over the importance ratios ω'_k of all SW layers in a CNN. The models are ResNet50 for ImageNet, ADE20K, and Cityscapes, ResNet44 for CIFAR-10, VGG16 for GTA5$\rightarrow$Cityscapes, and a ResNet-alike network for style transfer as in [15]. Both $\Omega = \{\text{bw}, \text{iw}\}$ and $\Omega = \{\text{bw}, \text{iw}, \text{bn}, \text{in}, \text{ln}\}$ are reported.

We make the following remarks: (1) For semantic segmentation, SW chooses mainly BW and BN, and partially the rest, while in classification more IW are selected. This is because the diversity between images is higher in classification datasets than in segmentation datasets. Thus more IW is required to alleviate the intra-dataset variance. (2) Semantic segmentation on Cityscapes tends to produce more IW and IN under domain adaptation setting than in the normal setting. Since domain adaptation introduces a domain discrepancy loss, more IW and IN would be beneficial for reducing the feature discrepancy between the two domains, *i.e.*, GTA5 and Cityscapes. (3) In image style transfer, SW switches to IW aggressively. This phenomenon is consistent with the common knowledge that IW is well suited for style transfer, as image level appearance information is well encoded in the covariance of CNN features. Our experiments also verify that IW is a better choice than IN in this task.

5. Conclusion

In this paper, we propose Switchable Whitening, which integrates various whitening and standardization techniques in a general form. SW adapts to various tasks by learning to select appropriate normalizers in different layers of a CNN. Our experiments show that SW achieves consistent improvements over previous normalization methods in a number of computer vision tasks, including classification, segmentation, domain adaptation, and image style transfer. Investigation of SW reveals the importance of leveraging different whitening methods in CNNs. We hope that our findings in this work would benefit other research fields and tasks that employ deep learning.

References

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 1, 2, 3

[2] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 11

[3] 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. *TPAMI*, 2018. 6

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CVPR*, 2016. 2, 5, 6

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009. 2, 5

[6] G. Desjardins, K. Simonyan, R. Pascanu, et al. Natural neural networks. *NIPS*, 2015. 2

[7] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 2012. 6

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *ICCV*, 2017. 11

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016. 2, 5

[10] L. Huang, Y. Zhou, F. Zhu, L. Liu, and L. Shao. Iterative normalization: Beyond standardization towards efficient whitening. *CVPR*, 2019. 4

[11] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *ICCV*, 2017. 2

[12] S. Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *NIPS*, 2017. 2, 3

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

[14] C. Ionescu, O. Vantzos, and C. Sminchisescu. Training deep networks with structured layers by matrix backpropagation. *ICCV*, 2015. 10

[15] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *ECCV*, 2016. 7, 8

[16] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Technical report, 2009. 2, 5

[17] H. Lei, Y. Dawei, L. Bo, and D. Jia. Decorrelated batch normalization. *CVPR*, 2018. 1, 2, 3, 4, 5, 10

[18] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos. Mmd gan: Towards deeper understanding of moment matching network. *NIPS*, 2017. 7

[19] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. *NIPS*, 2017. 1, 2, 3, 4

[20] X. Liang, H. Zhou, and E. Xing. Dynamic-structured semantic propagation network. *CVPR*, 2018. 6

[21] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. *CVPR*, 2017. 6

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. *ECCV*, 2014. 2, 7, 11

[23] P. Luo, Z. Peng, J. Ren, and R. Zhang. Do normalization layers in a deep convnet really need to be distinct? *arXiv preprint arXiv:1811.07727*, 2018. 5

[24] P. Luo, J. Ren, and Z. Peng. Differentiable learning-to-normalize via switchable normalization. *ICLR*, 2019. 2, 3, 11

[25] H. Nam and H.-E. Kim. Batch-instance normalization for adaptively style-invariant neural networks. *NIPS*, 2018. 2

[26] X. Pan, P. Luo, J. Shi, and X. Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. *ECCV*, 2018. 2, 10

[27] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. *ECCV*, 2016. 2, 6

[28] A. Siarohin, E. Sangineto, and N. Sebe. Whitening and coloring transform for gans. *arXiv preprint arXiv:1806.00420*, 2018. 2

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2, 6

[30] Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker. Learning to adapt structured output space for semantic segmentation. *CVPR*, 2018. 6

[31] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. *CVPR*, 2017. 1, 2, 3

[32] G. Wang, J. Peng, P. Luo, X. Wang, and L. Lin. Batch kalman normalization: Towards training deep neural networks with micro-batches. *NIPS*, 2018. 2

[33] Y. Wu and K. He. Group normalization. *ECCV*, 2018. 2, 3, 11

[34] Z. Wu, C. Shen, and A. Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 2019. 6

[35] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016. 6

[36] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context encoding for semantic segmentation. *CVPR*, 2018. 6

[37] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CVPR*, 2017. 5, 6

[38] H. Zhao, Y. Zhang, S. Liu, J. Shi, C. Change Loy, D. Lin, and J. Jia. Psanet: Point-wise spatial attention network for scene parsing. *ECCV*, 2018. 6

[39] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. *CVPR*, 2017. 2, 5

[40] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 2016. 6

Appendix

This appendix provides (1) back-propagation of SW, (2) discussion for our network configurations, (3) results for instance segmentation, and (4) some style transfer visualization results.

A. Back-propagation of SW

The backward pass of our proposed SW is presented in Algorithm 2.

Algorithm 2 Backward pass of SW for each iteration.

```
1: Input: mini-batch gradients respect to whitened outputs $\{\frac{\partial L}{\partial \hat{\mathbf{X}}_n}, n = 1, 2, ..., N\}$. Other auxiliary data from respective forward pass.
2: Output: the gradients respect to the inputs $\{\frac{\partial L}{\partial \mathbf{X}_n}, n = 1, 2, ..., N\}$; the gradients respect to the importance weights $\frac{\partial L}{\partial \lambda_k}$ and $\frac{\partial L}{\partial \lambda'_k}$, $k \in \{bw, iw\}$.
3: for $n = 1$ to $N$ do
4:     calculate $\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}_n}$ using results below.
5:     calculate $\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}_n}$ using results below.
6: end for
7: for $n = 1$ to $N$ do
8:     calculate $\frac{\partial L}{\partial \mathbf{X}_n} = \frac{\partial L}{\partial \hat{\mathbf{X}}_n}\mathbf{U}_n + (\frac{\omega_{bw}}{NHW}\sum_{i=1}^{N}\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}_i} + \frac{\omega_{iw}}{HW}\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}_n})$
       $+[\frac{2\omega'_{bw}(\mathbf{X_n} - \boldsymbol{\mu}_{bw})^T}{NHW}\sum_{i=1}^{N}(\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}_i})_{sym} + \frac{2\omega'_{iw}(\mathbf{X}_n - \boldsymbol{\mu}_{iw})^T}{HW}(\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}_n})_{sym}]$
9: end for
10: calculate:
    $\frac{\partial L}{\partial \lambda_{bw}} = \omega_{bw}(1-\omega_{bw})\sum_{n=1}^{N}(\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}_n}\boldsymbol{\mu}_{bw}) - \omega_{iw}\omega_{bw}\sum_{n=1}^{N}(\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}_n}\boldsymbol{\mu}_{iw}^{(n)})$
    $\frac{\partial L}{\partial \lambda_{iw}} = \omega_{iw}(1-\omega_{iw})\sum_{n=1}^{N}(\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}_n}\boldsymbol{\mu}_{iw}^{(n)}) - \omega_{bw}\omega_{iw}\sum_{n=1}^{N}(\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}_n}\boldsymbol{\mu}_{bw})$
    $\frac{\partial L}{\partial \lambda'_{bw}} = \omega'_{bw}(1-\omega'_{bw})\sum_{n=1}^{N}\langle\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}_n}, \boldsymbol{\Sigma}_{bw}\rangle_F - \omega'_{iw}\omega'_{bw}\sum_{n=1}^{N}\langle\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}_n}, \boldsymbol{\Sigma}_{iw}^{(n)}\rangle_F$ [1]
    $\frac{\partial L}{\partial \lambda'_{iw}} = \omega'_{iw}(1-\omega'_{iw})\sum_{n=1}^{N}\langle\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}_n}, \boldsymbol{\Sigma}_{iw}^{(n)}\rangle_F - \omega'_{bw}\omega'_{iw}\sum_{n=1}^{N}\langle\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}_n}, \boldsymbol{\Sigma}_{bw}\rangle_F$
```

In the Next, we derive $\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}_n}$ and $\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}_n}$ in the line 4, 5 of Algorithm 2, where we start with the forward pass of ZCA whitening.

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:

$$\hat{\boldsymbol{\Sigma}} = \mathbf{D}\boldsymbol{\Lambda}\mathbf{D}^T \tag{8}$$

$$\mathbf{V} = \boldsymbol{\Lambda}^{-1/2}\mathbf{D}^T \tag{9}$$

$$\tilde{\mathbf{X}} = \mathbf{V}(\mathbf{X} - \hat{\boldsymbol{\mu}} \cdot \mathbf{1}^T) \tag{10}$$

$$\hat{\mathbf{X}} = \mathbf{D}\tilde{\mathbf{X}} \tag{11}$$

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 [14, 17], $\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}}$ and $\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}}$ can be calculated as follows:

$$\frac{\partial L}{\partial \tilde{\mathbf{X}}} = \frac{\partial L}{\partial \hat{\mathbf{X}}}\mathbf{D} \tag{12}$$

$$\frac{\partial L}{\partial \mathbf{V}} = \frac{\partial L}{\partial \tilde{\mathbf{X}}}^T(\mathbf{X} - \hat{\boldsymbol{\mu}} \cdot \mathbf{1}^T)^T \tag{13}$$

$$\frac{\partial L}{\partial \boldsymbol{\Lambda}} = \frac{\partial L}{\partial \mathbf{V}}\mathbf{D}(-\frac{1}{2}\boldsymbol{\Lambda}^{-3/2}) \tag{14}$$

$$\frac{\partial L}{\partial \mathbf{D}} = \frac{\partial L}{\partial \mathbf{V}}\boldsymbol{\Lambda}^{-1/2}) + \frac{\partial L}{\partial \hat{\mathbf{X}}}\tilde{\mathbf{X}}^T \tag{15}$$

$$\frac{\partial L}{\partial \hat{\boldsymbol{\Sigma}}} = \mathbf{D}\{[\mathbf{K}^T \odot (\mathbf{D}^T\frac{\partial L}{\partial \mathbf{D}})] + (\frac{\partial L}{\partial \boldsymbol{\Lambda}})_{diag}\}\mathbf{D}^T \tag{16}$$

$$\frac{\partial L}{\partial \hat{\boldsymbol{\mu}}} = \frac{\partial L}{\partial \tilde{\mathbf{X}}}(-\mathbf{V}) \tag{17}$$

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.

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 [17] 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 [26] 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 to further save computation. Thus the normalization layers considered are the {1,6,12,...,36}*th* layers, containing 7 layers in total.

[1] $\langle\rangle_F$ denotes Frobenius inner product.

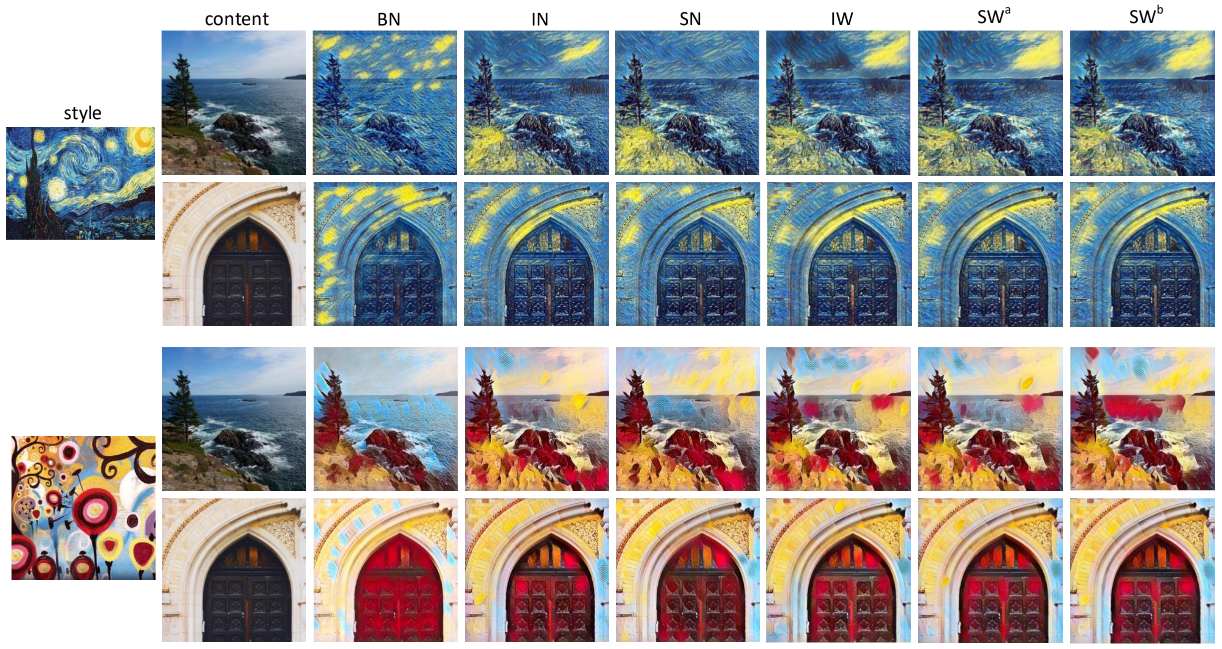


Figure 9. Visualization of style transfer using different normalization layers.

Table 7. Mask R-CNN using ResNet50 and FPN with 2× LR schedule.

Backbone	FPN & Head	AP$_{box}$	AP$_{mask}$
FrozenBN	-	38.5	35.1
SyncBN	SyncBN	39.6	35.6
GN	GN	39.6	35.8
SN	SN	41.0	36.5
SW^a	SyncBN	**41.2**	**37.0**

C. Instance Segmentation

We further provide results on instance segmentation, where Mask-RCNN [8] and COCO dataset [22] are used to evaluate our method, and the implementation is based on mmdetection [2]. We replace 7 normalization layers of the ResNet50 backbone with SW following the same way as in ImageNet, while the rest normalization layers of the backbone, FPN, and detection/mask head are SyncBN. The SW layers are synchronized across multiple GPUs. As shown in Fig.7, SW significantly outperforms SyncBN and GN [33], and also outperforms SN reported by [24], which replaces all normalization layers to SN.

D. Style Transfer Results

Fig.9 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.