

WeightAlign: Normalizing Activations by Weight Alignment

Xiangwei Shi*, Yunqiang Li*, Xin Liu* and Jan van Gemert
Computer Vision Lab
Delft University of Technology, The Netherlands

Abstract—Batch normalization (BN) allows training very deep networks by normalizing activations by mini-batch sample statistics which renders BN unstable for small batch sizes. Current small-batch solutions such as Instance Norm, Layer Norm, and Group Norm use channel statistics which can be computed even for a single sample. Such methods are less stable than BN as they critically depend on the statistics of a single input sample. To address this problem, we propose a normalization of activation without sample statistics. We present WeightAlign: a method that normalizes the weights by the mean and scaled standard derivation computed within a filter, which normalizes activations without computing any sample statistics. Our proposed method is independent of batch size and stable over a wide range of batch sizes. Because weight statistics are orthogonal to sample statistics, we can directly combine WeightAlign with any method for activation normalization. We experimentally demonstrate these benefits for classification on CIFAR-10, CIFAR-100, ImageNet, for semantic segmentation on PASCAL VOC 2012 and for domain adaptation on Office-31.

I. INTRODUCTION

Batch Normalization [1] is widely used in deep learning. Examples include image classification [2], [3], [4], object detection [5], [6], [7], semantic segmentation [8], [9], [10], generative models [11], [12], [13], *etc.* It is fair to say that for optimizing deep networks, BatchNorm is truly the norm.

BatchNorm stabilizes network optimization by normalizing the activations during training and exploits mini-batch sample statistics. The performance of the normalization thus depends critically on the quality of these sample statistics. Having accurate sample statistics, however, is not possible in all applications. An example is domain adaptation, where the statistics of the training domain samples do match the target domain statistics, and alternatives to BatchNorm are used. [14], [15]. High resolution images are another example, as used in object detection [16], [5], [7], segmentation [8], [9] and video recognition [17], [18], where only a few or just one sample per mini-batch fits in memory. For these cases, it is difficult to accurately estimate activation normalization statistics from the training samples.

To overcome the problem of unreliable sample statistics, various normalization techniques have been proposed which make use of other statistics derived from samples, such as layer [19], instance [20] or group [21]. These methods are

applied independently per sample and achieve good performance, but gather statistics just on a single sample and are thus less reliable than BatchNorm.

In this paper, we propose WeightAlign: normalizing activations without using sample statistics. Instead of sample statistics, we re-parameterize the weights within a filter to arrive at correctly normalized activations. See Fig. 1 for a visual overview. Our method is based on weight statistics and is thus orthogonal to sample statistics. This allows us to exploit two orthogonal sources to normalize activations: the traditional one based on sample statistics in combination with our proposed new one based on weight statistics. We have the following contributions.

- WeightAlign: A new method to normalize filter weights.
- Activation normalization without computing sample statistics.
- Performance independent of batch size, and stable performance over a wide range of batch sizes.
- State-of-the-art performance on 5 datasets in image classification, object detection, segmentation and domain adaptation.

II. RELATED WORK

Network Normalization by Sample Statistics. The idea of whitening the input to improve training speed [22] can be extended beyond the input layer to intermediate representations inside the network. Local Response Normalization [4], used in AlexNet, normalizes the activations in a small neighborhood for each pixel. Batch Normalization (BN) [1] performs normalization using sample statistics computed over mini-batch, which is helpful for training very deep networks. BN unfortunately suffers from performance degradation when the statistical estimates become unstable for small batch-size based tasks.

To alleviate the small batches issue in BN, Batch Renormalization [23] introduces two extra parameters to correct the statistics during training. However, Batch Renormalization is still dependent on mini-batch statistics, and degrades performance for smaller mini-batches. EvalNorm [24] corrects the batch statistics during inference procedure. But it fails to fully alleviate the small batch issue. SyncBN [25] handles the small batch problem by computing the mean and variance across multiple GPUs which is not possible without having multiple GPUs available.

* The authors contributed equally to this work.

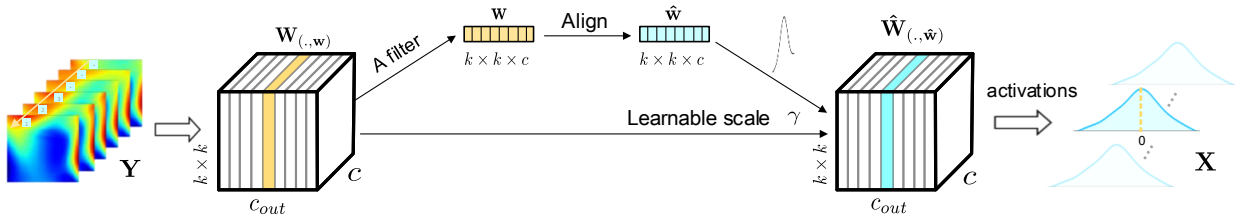


Fig. 1. Overview of WeightAlign (WA): Aligning filter weights allows normalizing channel activations. The weight \mathbf{W} of an arbitrary convolutional layer is composed with C_{out} filters with size $C \times k \times k$. Within a filter (yellow region in the left side), we first flatten it to a vector and then apply our proposed WA in Eq. (10) to normalize the filter weights with a introduced learnable parameter γ . Normalizing weights of each filter in forward pass can realize the normalization of activations within each channel. Details in Section III.

Because small mini-batch sample statistics are unreliable, several methods [19], [20], [21] perform feature normalization based on channel sample statistics. Instance Normalization [20] performs normalization similar to BN but only for a single sample. Layer Normalization [19] uses activation normalization along the channel dimension for each sample. Filter Response Normalization [26] proposes a novel combination of a normalization that operates on each activation channel of each batch element independently. Group Normalization [21] divides channels into groups and computes the mean and variance within each group for normalization. Local Context Normalization [27] normalizes every feature based on the filters in its group and a window around it. These methods can alleviate the small batch problem to some extent, yet have to compute statistics for just a single sample both during training and inference which introduces instabilities [28], [29]. Instead, in this paper we cast the activation normalization over *feature space* into weights manipulation over *parameter space*. Network parameters are independent of any sample statistics and makes our method particularly well suited as an orthogonal information source which can compensate for unstable sample normalization methods.

Importance of weights. Proper network weight parameter initialization [30], [4], [31] is essential for avoiding vanishing and exploding gradients [32]. Randomly initializing weight values from a normal distribution [4] is not ideal because of the stacking of non-linear activation functions. Properly scaling the initialization for stacked sigmoid or tanh activation functions leads to Xavier initialization [33], but it is not valid for ReLU activations. He *et al.* [3] extend Xavier initialization [33] and derive a sound initialization for ReLU activations. For hypernets, [34] developed principled techniques for weight initialization to solve the exploding activations even with the use of BN. In [35] a data-dependent initialization is proposed to mimic the effect of BN to normalize the variance per layer to one in the first forward pass. To address the initialization problem of residual nets, Zhang *et al.* [36] propose fixup initialization. We draw inspiration from these weight initialization methods to derive our weight alignment for activation normalization.

Weight decay [37] adds a regularization term to encourage small weights. Max-norm regularization is used in [38] to

avoid extremely large weights. Instead of introducing an extra term to the loss, Weight Norm [39] re-parameterizes weights by dividing its norm to accelerate convergence. Such a re-parameterization, however, may greatly magnify the magnitudes of input signals. In contrast, we scale each filter's weights as inspired by He *et al.* [3]'s weight initialization approach, to generate activations with zero mean and maintained variance. This allows us to normalize activations without using any sample statistics.

III. PROPOSED METHOD

Batch Normalization (BN) [1] normalizes the features in a single channel to a zero mean and unit variance distribution, as in Fig. 2, to stabilize the optimization of deep networks. The normalized features $\hat{\mathbf{x}}$ are computed channel-wise at training time using the sample mean μ_β and standard derivation σ_β over the input features \mathbf{x} as:

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \mu_\beta}{\sigma_\beta}, \quad (1)$$

where μ_β and σ_β are functions of *sample statistics* of \mathbf{x} . For each activation $\hat{\mathbf{x}}$, a pair of trainable parameters γ , β are introduced to scale and shift the normalized value that

$$\mathbf{r} = \gamma \hat{\mathbf{x}} + \beta. \quad (2)$$

BN uses the running averages of sample statistics within each mini-batch to reflect the statistics over the full training set. A small batch leads to inaccurate estimation of the sample statistics, thus using small batches degrades the accuracy severely.

In the following, we express the sample statistics in terms of filter weights to eliminate the effect of small batch, and re-parameterize the weights within each filter to realize the normalization of activations within each channel.

A. Expressing activation statistics via weights

For a convolution layer, let \mathbf{x} be a single channel output activation, \mathbf{Y} be the input activations, and \mathbf{w} , b be the corresponding filter and bias scalar. Specifically, a filter \mathbf{w} consists of c channels with a spatial size of $k \times k$. To normalize activations by using weights, we show how the sample statistics μ_β and σ_β in Eq. (1) are expressed in terms of filter weights.

An individual response value $x(t)$ in \mathbf{x} is a sum of the product between the values in \mathbf{w} and the values in a co-located k^2c -by-1 vector $\mathbf{y}(t)$ in \mathbf{Y} , which is equivalently written as the dot product of filter \mathbf{w} and vector $\mathbf{y}(t)$,

$$x(t) = \mathbf{w} \cdot \mathbf{y}(t) + b. \quad (3)$$

By computing each individual response $x(t)$ at any location t , we obtain the full single channel output activation \mathbf{x} .

We use random variables x , Y and w to present each of the elements in \mathbf{x} , \mathbf{Y} and \mathbf{w} respectively. Following [34], [33], [3], we make the following assumptions about the network: (1) The w , Y and b are all independent of each other. (2) The bias term b equals 0.

Expressing μ_β of Eq. (1) in terms of weights: The mean of activation \mathbf{x} can be equivalently represented via filter weights as:

$$\mu_\beta = \mathbb{E}[x] = n\mathbb{E}[wY] = n\mathbb{E}[w]\mathbb{E}[Y], \quad (4)$$

where \mathbb{E} is the expected value and $n = k^2c$ denotes the number of weight values in a filter.

Expressing σ_β of Eq. (1) in terms of weights: The variance of activation \mathbf{x} is:

$$\begin{aligned} \sigma_\beta^2 &= \text{Var}[x] = n\text{Var}[wY] = n(\mathbb{E}[w^2Y^2] - \mathbb{E}^2[wY]) \\ &= n(\mathbb{E}[w^2]\mathbb{E}[Y^2] - \mathbb{E}^2[w]\mathbb{E}^2[Y]) \end{aligned} \quad (5)$$

With the Eq. (4) and Eq. (5), we will manipulate the weights to normalize activations in the following sections.

B. WeightAlign

We normalize activations by encouraging the mean μ_β of the activations in Eq. (4) to be zero, and maintaining the variance of the activations to be the same in all layers. Inspired by [3], we manipulate the weights to satisfy these requirements.

The mean of the activations μ_β in Eq. (4) is forced to be zero when the weight w in a filter has zero mean:

$$\mathbb{E}[w] = 0. \quad (6)$$

We can exploit $\mathbb{E}[w] = 0$ in Eq. (5) to further simplify the variance of the activations σ_β^2 in terms of the variance of the weights as,

$$\sigma_\beta^2 = \text{Var}[x] = n\mathbb{E}[w^2]\mathbb{E}[Y^2] = n\text{Var}[w]\mathbb{E}[Y^2]. \quad (7)$$

Thus, the activation variance σ_β^2 depends on the variance of the weights w and on the current layer input Y . To simplify further, we follow [3] and assume a ReLU activations function. We use Z to define the outputs of the previous layer before the activation function, where $Y = \text{ReLU}(Z)$.

If the weight of the previous layer has a symmetric distribution around zero then Z has a symmetric distribution around zero.¹ Because the ReLU sets all negative values to 0, the variance is halved [3], and we have that $\mathbb{E}[Y^2] = \frac{1}{2}\text{Var}[Z]$. Substituted into Eq. (7), we have

$$\sigma_\beta^2 = \text{Var}[x] = \frac{1}{2}n\text{Var}[w]\text{Var}[Z]. \quad (8)$$

This shows the relationship of variance between the previous layer's output before the non-linearity Z and the current layer's single channel activation x .

Our goal is to manipulate the weights to keep the activation variance of all layers normalized. Thus we relate the variance of this layer σ_β^2 with the variance in the previous layer $\text{Var}[Z]$, which can be achieved in terms of the weights by setting $\frac{1}{2}n\text{Var}[w]$ in Eq. (8) to a proper scalar (e.g. 1):

$$\frac{1}{2}n\text{Var}[w] = 1. \quad (9)$$

With Eq. (6) and Eq. (9), we can re-parameterize a single filter weights to have zero mean and a standard deviation $\sqrt{2/n}$, which achieves the activation normalization like BN in Eq. (1). Thus, here we propose WeightAlign as,

$$\hat{w} = \gamma \frac{w - \mathbb{E}[w]}{\sqrt{n/2 \cdot \text{Var}[w]}}, \quad (10)$$

where γ is a learnable scalar parameter similar to the scale parameter in BN [1] like Eq. (2). Since WA manipulates the weights, it does not have the β term as in the form of BN, which is applied on activations directly. The detail explanation about this is given in supplementary material. From Eq. (10), we can tell that the proposed WeightAlign(WA) method does not rely on sample statistics computed over a mini-batch, which makes it independent of batch size.

C. Initialization of weights

We initialize each filter weights \mathbf{w} to be a zero-mean symmetric Gaussian distribution whose standard deviation (std) is $\sqrt{2/n}$ where $n = k^2c$. For the first layer, we should have $\sqrt{1/n}$ since the ReLU activation is not applied on the input. But the factor 1/2 can be neglected if it just exists on one layer and for simplicity, we use the same deviation $\sqrt{2/n}$ for initialization.

D. Empirical analysis and examples

To show the distributions of channel activations using different methods, we build an 8-layer deep network containing 7 convolutional layers and one classification layer with ReLU as nonlinear activation function. A mini-batch of size 128 independent data is sampled from $\mathcal{N}(0, 1)$ is used as input. Specifically, the standard Kaiming initialization [3] without bias term is used to initialize the weights. Fig. 2 shows the activation distributions of 8 different channels taken from two layers, the 3rd intermediate convolutional layer and the last classifier layer before the softmax loss. Each channel of the last classifier layer represents one class. To exclude the training effect, we first use the initialization model to conduct comparison between sample statistics based normalization and our WA normalization. For the visualization of trained model, please see Section IV.

For the baseline model, it can be seen in Fig. 2(a) that the activation distributions in the intermediate layer start drifting, leading to a constant output for the classifier layer (i.e. the 'Blue' indexed class/channel). From Fig. 2(b) of Batch

¹ See proof in the supplementary material.

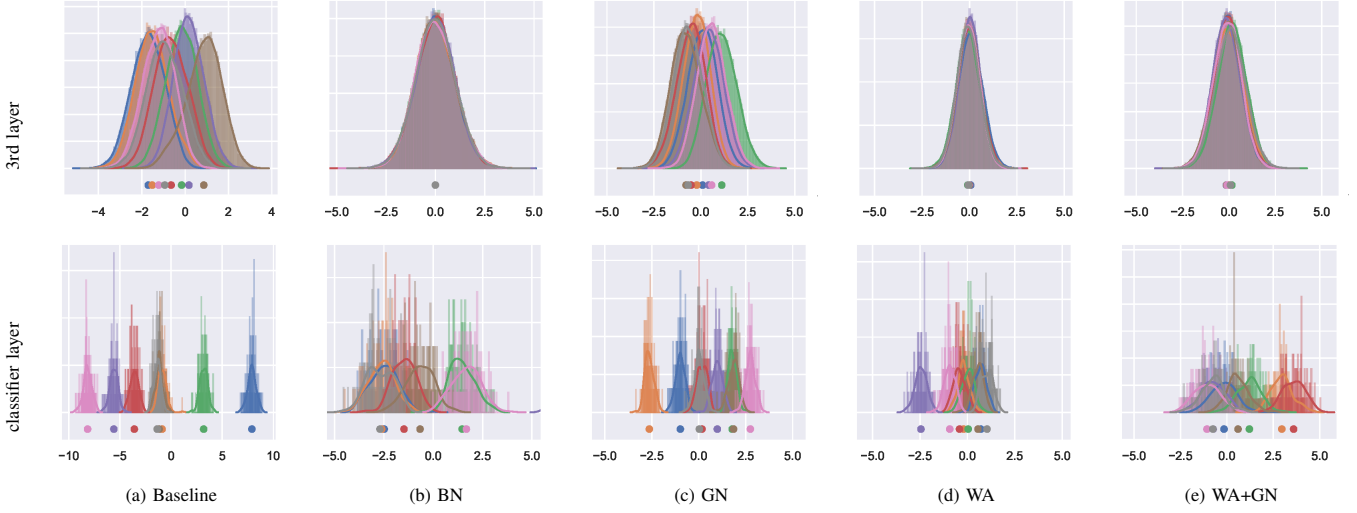


Fig. 2. Our WA method can alleviate internal covariate shift of activation (see Section III-D for details). Each color represents the activation distribution of different channels for different layers. For baseline model, the ‘Blue’ indexed channel will dominate all other channels, leading to a constant classification result. Normalizing channel activation in intermediate layer can avoid the constant output for the final classifier layer, which can be realized by WA.

Norm (BN) [1], we note that reducing the Internal Covariate Shift (ICS) in intermediate layer can alleviate the distribution drifting for the final classifier layer. The output of the classifier layer is no longer a constant function. Group Norm(GN) [21] can also reduce ICS to some extent, as shown in Fig. 2(c).

As shown in Fig. 2(d), WA can realize similar functionality of BN. Our proposed WeighAlign (WA) re-parameterizes the weights within each filter in Eq. (10) to normalize channel activation. Since weight statistics is orthogonal to sample statistics, WA can be used in conjunction with BN, GN, LN and IN, as shown in Fig. 2(e). Please refer to supplementary material for visualization of other normalization methods.

IV. EXPERIMENTS

We show extensive experiments on three tasks and five datasets: image classification on CIFAR-10, CIFAR-100 [40] and ImageNet; domain adaptation on Office-31; and semantic segmentation on PASCAL VOC 2012 [41] where we evaluate VGG [31] and residual networks as ResNet [30].

A. Datasets

CIFAR-10 & CIFAR-100. CIFAR-10 and CIFAR-100 [40] consist of 60,000 32×32 color images with 10 and 100 classes, respectively. For both datasets, 10,000 images are selected as the test set and the remaining 50,000 images are used for training. We perform image classification experiments and ablation study on these datasets.

ImageNet. ImageNet [42] is a classification dataset with 1,000 classes. The size of the training dataset is around 1.28 million, and 50,000 validation images are used for evaluation.

Office-31. Office-31 [43] is a dataset for domain adaptation with 4,652 images with 31 categories. The images are collected from three distinct domains: Amazon (A), DSLR (D), and Webcam (W). The largest domain, Amazon, has 2,817 labeled images. The 31 classes consists of objects commonly

encountered in office settings, such as file cabinets, keyboards and laptops.

PASCAL VOC 2012. PASCAL VOC 2012 [41] contains a semantic segmentation set and includes 20 foreground classes and a single background class. The original segmentation dataset contains 1,464 images for training, and 1,499 images for evaluation. We use the augmented training dataset including 10,582 images [44].

B. Implementation details

We use Stochastic Gradient Descent(SGD) in all experiments with a momentum of 0.9 and a weight decay of 5×10^{-4} . The plain VGG model is initialized with Kaiming initialization [3] and ResNets are initialized with Fixup [36]. We do not apply WA in the final classifier layer. For classification, we train the models with data augmentation, random horizontal flipping and random crop, as in [45]. Further implementation details are given in each subsection.

C. Experiments on CIFAR-10 & CIFAR-100

Comparing normalization methods. We compare our proposed WeightAlign (WA) against various activation normalization methods, including Batch Normalization (BN) [1], Group Normalization (GN) [21], Layer Normalization (LN) [19], Instance Normalization (IN) [20]. To demonstrate that WA is orthogonal to all these approaches, we also show the combination of WA with these activation normalization methods.

We conduct experiments on CIFAR-100 as shown in Table I, where all models are trained with a batch size of 64. WeightAlign outperforms every normalization method except BN over large batch size. By combining WeightAlign with sample statistics normalization methods, we can see that weightAlign adds additional information and improves accuracy. Especially, the GN+WA model achieves comparable performance of BN.

To further show it’s flexibility, we fine-tune a pre-trained BN model on CIFAR-100 with WA. It achieves 22.38% error

TABLE I
ERROR RATE OF RESNET50 ON CIFAR-100 FOR CLASSIFICATION. WA OUTPERFORMS EVERY NORMALIZATION METHOD EXCEPT BN OVERLARGE BATCH SIZE. WHEN COMBINED WITH STATISTICS NORMALIZATION METHODS, WA IMPROVES ACCURACY.

CIFAR-100			
Method	Error	Method	error
Baseline [36]	28.43	WA	24.92
BN [1]	23.02	BN + WA	22.39
IN [20]	25.58	IN + WA	24.63
LN [19]	26.78	LN + WA	24.01
GN [21]	25.46	GN + WA	23.64

TABLE II
ERROR RATE OF RESNET50 ON CIFAR-10 FOR CLASSIFICATION. WEIGHT STATISTICS NORMALIZATION (WA) HELPS IN ADDITION TO SAMPLE STATISTICS NORMALIZATION.

CIFAR-10							
Batch size 64				Batch size 1			
Method	Error	Method	Error	Method	Error	Method	Error
Baseline [36]	6.46	WA	6.21	Baseline	7.27	WA	6.61
BN [23]	4.30	BN+WA	4.29	BN	-	BN+WA	-
IN [20]	6.49	IN+WA	6.42	IN	6.91	IN+WA	6.50
LN [19]	5.02	LN+WA	5.12	LN	6.82	LN+WA	5.76
GN [21]	4.96	GN+WA	4.60	GN	5.79	GN+WA	5.51

rate, which is the same as BN+WA model trained from scratch. This shows the possibility of finetuning WA on a pre-trained models with a sample-based normalization method.

On CIFAR-10 we do the same evaluation for ResNet50. BN and BN+WA are trained only with batch size of 64, and other methods are trained with batch size of 1 and 64. For batch size of 64 and 1, the initial learning rates are 0.01 and 0.001, respectively. The baseline method [36] is trained without any normalization methods. Results are given in Table II. Because CIFAR-10 is less complex than CIFAR-100, the performance gaps between different methods are thus less obvious. Nevertheless, our method is comparable to other normalization methods and it allows a batch size of 1, which is not possible for BN. By combining WA with the sample-based normalization methods, we again observe that weight statistics normalization helps in addition to sample statistics normalization.

We also compare WA with Weight Normalization (WN) [39] on CIFAR-100 with VGG16 and ResNet50 networks. For VGG16, the error rates of WN, WA, GN are 6.8%, 4.8% and 7.4% when the batch size is 1, and the ones of WN, WA, BN, GN are 8.6%, 5.4%, 5.8% and 11.0% when the batch size is 64. For ResNet50, WN cannot be trained. We can see WA and WN are workable regardless of the batch size for plain networks and WA outperforms WN in this model. For residual networks, WA can work alone without any activation normalization layer, while WN cannot. WN is only normalized by its norm to accelerate training. Without normalization layer in residual networks, WN cannot restrict

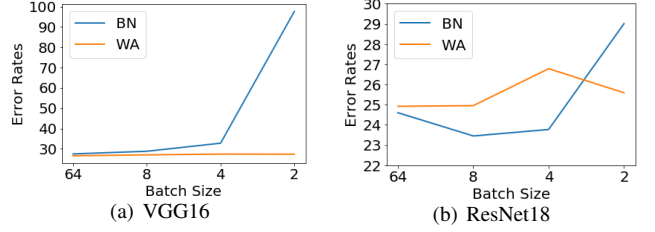


Fig. 3. Classification error vs. batch sizes. The models are VGG16 and ResNet18 trained on CIFAR-100. The error rates of BN increase rapidly when small batch sizes are used. Our proposed method does not rely on the sample statistics and is independent of batch size dimension. The error rates of WA are stable over a wide range of batch sizes.

the variance of the activation and the residual structure merges the activation from two branches, which leads to exploding gradients problem. Instead, WA normalizes the weights by zero-mean and a scaled standard deviation. This scale is determined by the size of convolutional filter and restricts the variance of weights and further maintains the variance of activations.

Small batch sizes. In Fig. 3, we compare BN and WA with different batch sizes on VGG16 and ResNet18. We train with batch sizes of 64, 8, 4, 2 images per GPU. In all cases, the means and variances in the batch normalization layers are computed within each GPU and not synchronized. The x-axis shows different batch sizes and y-axis presents the validation error rates. From Fig. 3 (a), we observe that the performance of WA is comparable to that of BN on the plain network with large batches. The error rates of BN increase rapidly when reducing the batch size, especially with batch size smaller than 4. In contrast, our method focuses on the normalization of weight distributions which is independent of batch size. The error rates of our method are stable over different batch sizes. We can observe similar results in Fig. 3 (b) that our method performs stable over different batch size on residual network. It is interesting to see that the performance gap between WA and BN with ResNet18 is smaller than that with VGG16, which is due to the residual structure.

Depth of residual networks. We evaluate depth of 18, 34, 50, 101, 152 for residual networks with batch sizes of 1, 64 per GPU. Table III shows the validation error rates on CIFAR-10. We observe that WA is able to train very deep networks for both small and large batch sizes.

Visualization of weights and activations in training. To verify the motivation, in this subsection, we visualize the distribution of weights and activation by training a ResNet50 model on CIFAR-100. A minibatch of 128 images from CIFAR-100 is input to the trained network to obtain features and weights.

How are the filter weights distributed? In Fig. 4, we visualize how the channel weight distribution changes with the epochs. Fig. 4 (a) illustrates the baseline ResNet50 model without using any normalization. When using BN, as shown in

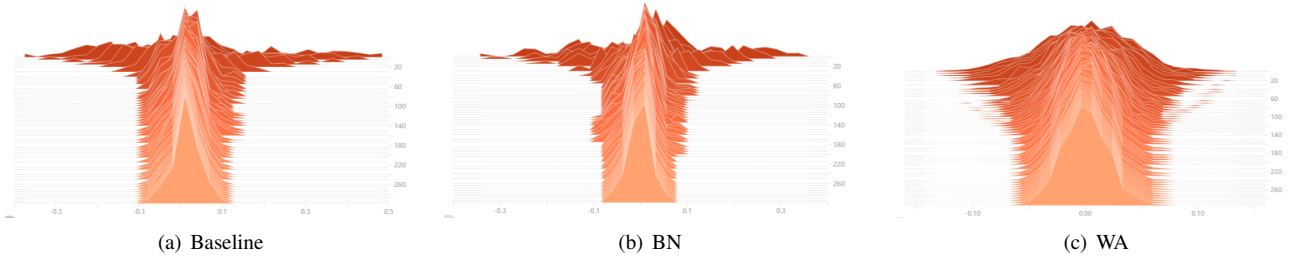


Fig. 4. Channel weights distribution of trained ResNet50 versus epoch: (a) channel weights of baseline model. (b) channel weights of BN model. (c) channel weights of WA model. The x-axis represents the value of weights. The y-axis represents the epoch. From top to bottom, the epoch varies from 1 to 300. WA makes the distributions of channel weights symmetric around 0. With a symmetric distribution around zero for the weight, the activation can also have a symmetric distribution around zero.

TABLE III

APPLYING WA TO VARYING DEPTH MEASURED BY ERROR RATE. WA IS ABLE TO TRAIN VERY DEEP NETWORKS FOR BOTH SMALL AND LARGE BATCH SIZES.

CIFAR-10		
Model (+WA)	Batchsize 1	Batchsize 64
	Error	Error
ResNet18	5.65	6.23
ResNet34	5.84	5.78
ResNet50	6.61	6.42
ResNet101	5.74	6.41
ResNet152	6.09	6.52

TABLE IV

COMPARATIVE RESULTS FOR VGG16 AND RESNET18 MODELS WITH DIFFERENT COMPONENTS. BOTH COMPONENTS CONTRIBUTE SUBSTANTIALLY TO THE PERFORMANCE.

Components in WA		VGG16	ResNet18
$\mathbb{E}[w] = 0$	$\text{Var}[w] = \frac{2}{n}$	Error	Error
×	×	-	28.23
✓	×	35.74	27.97
×	✓	-	28.03
✓	✓	27.85	24.92

Fig. 4 (b), the weight distribution tends to be symmetric around zero. From Fig. 4 (c), we note that the proposed WA method can realize the same functionality of BN making the weights symmetric and smooth during training. Channel and layer weights distribution during training of other normalization methods can be found in the supplementary material. With a symmetric distribution around zero for the weight, the activation can also have a symmetric distribution around zero.

How are the channel activations distributed? In Fig. 2 we have found that before training, the proposed WA method can effectively normalize channel activations by aligning the weights within each filter. In Fig. 5 we further explore how the channel activations distributed on a trained ResNet50 network. We visualize the activation distributions for 4 different channels taken from two layers, a middle layer (the 2rd conv layer of the 8th residual block) and the last classifier layer. Since the weight decay is used, the scale and shift parameters γ, β are converged to small values. From Fig. 5 (d), we have similar observations as in Fig. 2 that our method can normalize activation of different channels to zero mean and same variance. Fig. 5 (e) also illustrates WA can be used in conjunction with GN to further normalize the activations.

Functionality of different components. We align the filter weights by subtracting the mean and dividing by a properly scaled derivation. For our method, there are two components: $\mathbb{E}[w] = 0$ in Eq. (6) $\text{Var}[w] = \frac{2}{n}$ in Eq. (9). We conduct an comparative results for our model with

different components for VGG and ResNet18 models on CIFAR-100. In Table IV we start with the baseline model [36] and augment incrementally with the two components. We observe that both components contribute substantially to the performance of the whole model.

D. Comparison with state-of-the-art

Image classification on ImageNet. We experiment with a regular batch size of 64 images per GPU on plain and residual networks. The total training epoch is 100. The initial learning rate is 0.1 and divided by 10 at the 30th, 60th and 90th epoch. Table V shows the top-1 and top-5 error rates of image classification on ImageNet. For plain networks,

TABLE V

TOP-1 AND TOP-5 ERROR RATES OF IMAGE CLASSIFICATION ON IMAGENET. WA AND WA+BN CAN WORK WELL ON LARGE AND COMPLEX DATASETS.

model	Top-1 (%) Error	Top-5 (%) Error
VGG16* (Baseline)	31.30	11.19
VGG16 (BN)*	29.58	10.16
VGG16 (WA)	29.78	10.23
VGG16 (BN+WA)	27.07	8.78
ResNet50 (Baseline) [†] [36]	27.60	-
ResNet50 (BN)*	24.89	7.71
ResNet50 (WA)	26.62	8.91
ResNet50 (BN+WA)	24.04	7.12

The * denotes that we directly test the PyTorch pretrained model. The [†] denotes the numbers from the reference.

WA performs relatively well as BN, only 0.2% difference on

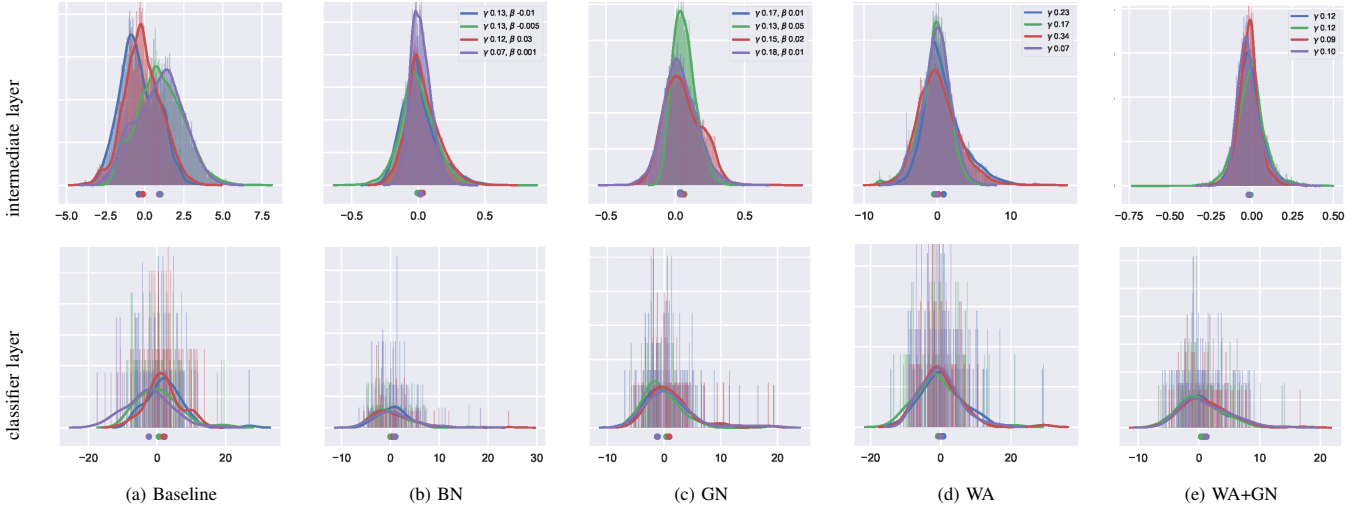


Fig. 5. Channel activation distributions of trained models on CIFAR-100. Different color represents different channel. γ and β are the parameters in each normalization. WA can align the channel activations by normalizing channel weights.

top-1 error rate. Combining WA with BN outperforms BN on VGG16 by 2.51% top-1 error rate and 1.38% top-5 error rate. For residual networks, WA performs slightly worse than BN, but combined BN+WA surpasses BN by 0.85% on top-1 error rate and 0.59% on top-5 error rate. This shows that WA and WA+BN can work well on large and complex datasets.

Domain adaptation on Office-31. Table VI presents overall scores of different normalization methods and their combination with WA on Office-31 dataset. Here we use ResNet50 models pretrained on CIFAR-100, which are obtained from previous experiments. To perform domain adaptation experiments, we finetune the pretrained model on a source domain, and validate it on two target domains with 0.001 initial learning rate, e.g. finetune on A, and validate on W and D. Model with WA consistently outperforms model with GN, IN and LN. Table VI also shows that our method WA can be applied to other normalization methods successfully and improves domain adaptation performance. In Office-31, the three domains A, W and D have different sample distribution. Thus the normalization methods that rely on sample statistics perform worse than our method WA, which only applies on weights and is independent of sample statistics. When combined with WA, other activation normalization methods can benefit from the sample-independent merit of WA, which boosts their domain adaptation performance.

Semantic segmentation on PASCAL VOC 2012. To investigate the effect on small batch size, we conduct experiments on semantic segmentation with PASCAL VOC 2012 dataset. We select the DeepLabv3 [46] as baseline model in which the ResNet50 pre-trained on ImageNet is used as backbone. Given the pretrained ResNet50 models with BN and BN+WA on ImageNet, we finetune them on PASCAL VOC 2012 for 50 epochs with batch size 4, 0.007 learning rate with polynomial decay and multi-grid (1,1,1),

TABLE VI
CLASSIFICATION ACCURACIES (%) ON OFFICE-31 DATASET (RESNET50).
WA CAN BE APPLIED TO OTHER NORMALIZATION METHODS AND
IMPROVES DOMAIN ADAPTATION PERFORMANCE.

Method	A \rightarrow W	W \rightarrow A	A \rightarrow D	D \rightarrow A	W \rightarrow D	D \rightarrow W	Avg
WA	34.62	42.89	52.05	44.09	87.67	80.00	56.89
BN [1]	46.92	48.91	57.53	37.11	90.41	70.00	58.48
BN + WA	49.23	36.38	58.90	40.00	91.78	76.92	58.87
GN [21]	46.92	42.89	50.68	39.76	82.19	75.38	56.30
GN + WA	46.15	54.46	50.68	37.83	87.67	78.46	59.21
IN [20]	36.92	33.98	41.10	37.11	84.93	76.92	51.83
IN + WA	39.23	41.45	38.36	38.07	90.41	73.85	53.56
LN [19]	37.69	29.89	45.20	33.97	79.45	69.23	49.24
LN + WA	50.77	42.89	56.16	41.69	91.78	80.77	60.68

and evaluate the performance on the validation set with output stride 16. The evaluation mIoU of BN method is 73.80%, and that of BN+WA is 74.87%. Combined with WA, the original BN performance is improved by 1.03%. This further shows that WA improves other normalization methods.

V. CONCLUSION

We propose WeightAlign; a method that re-parameterizes the weights by the mean and scaled standard derivation computed within a filter. We experimentally demonstrate WeightAlign on five different datasets. WeightAlign does not rely on the sample statistics, and performs on par with Batch Normalization regardless of batch size. WeightAlign can also be combined with other activation normalization methods and consistently improves on Batch Normalization, Group Normalization, Layer Normalization and Instance Normalization on various tasks, such as image classification, domain adaptation and semantic segmentation.

REFERENCES

- [1] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015. 1, 2, 3, 4, 5, 7
- [2] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" *IEEE transactions on image processing*, vol. 24, no. 12, pp. 5017–5032, 2015. 1
- [3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034. 1, 2, 3, 4
- [4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. 1, 2
- [5] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969. 1
- [6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788. 1
- [7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99. 1
- [8] 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*, 2015, pp. 3431–3440. 1
- [9] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528. 1
- [10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241. 1
- [11] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017. 1
- [12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680. 1
- [13] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015. 1
- [14] W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han, "Domain-specific batch normalization for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7354–7362. 1
- [15] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, "Revisiting batch normalization for practical domain adaptation," *arXiv preprint arXiv:1603.04779*, 2016. 1
- [16] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448. 1
- [17] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308. 1
- [18] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497. 1
- [19] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016. 1, 2, 4, 5, 7
- [20] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016. 1, 2, 4, 5, 7
- [21] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19. 1, 2, 4, 5, 7
- [22] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48. 1
- [23] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," in *Advances in neural information processing systems*, 2017, pp. 1945–1953. 1, 5
- [24] S. Singh and A. Shrivastava, "Evalnorm: Estimating batch normalization statistics for evaluation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3633–3641. 1
- [25] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "Megdet: A large mini-batch object detector," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6181–6189. 1
- [26] S. Singh and S. Krishnan, "Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 237–11 246. 2
- [27] A. Ortiz, C. Robinson, D. Morris, O. Fuentes, C. Kiekintveld, M. M. Hassan, and N. Jojic, "Local context normalization: Revisiting local normalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 276–11 285. 2
- [28] W. Shao, T. Meng, J. Li, R. Zhang, Y. Li, X. Wang, and P. Luo, "Ssn: Learning sparse switchable normalization via sparsestmax," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 443–451. 2
- [29] J. Yan, R. Wan, X. Zhang, W. Zhang, Y. Wei, and J. Sun, "Towards stabilizing batch statistics in backward propagation of batch normalization," *ICLR*, 2020. 2
- [30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 2, 4
- [31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 2, 4
- [32] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318. 2
- [33] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256. 2, 3
- [34] O. Chang, L. Flokas, and H. Lipson, "Principled weight initialization for hypernetworks," in *International Conference on Learning Representations*, 2020. 2, 3
- [35] D. Mishkin and J. Matas, "All you need is a good init," *arXiv preprint arXiv:1511.06422*, 2015. 2
- [36] H. Zhang, Y. N. Dauphin, and T. Ma, "Fixup initialization: Residual learning without normalization," *arXiv preprint arXiv:1901.09321*, 2019. 2, 4, 5, 6
- [37] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in neural information processing systems*, 1992, pp. 950–957. 2
- [38] N. Srebro and A. Shraibman, "Rank, trace-norm and max-norm," in *International Conference on Computational Learning Theory*. Springer, 2005, pp. 545–560. 2
- [39] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 901–909. 2, 5
- [40] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009. 4
- [41] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge 2007 (voc2007) results," 2007. 4
- [42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255. 4
- [43] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proceedings of the 11th European Conference on Computer Vision: Part IV, ser. ECCV'10*. Berlin, Heidelberg: Springer-Verlag, 2010, p. 213–226. 4
- [44] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 991–998. 4
- [45] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *arXiv preprint arXiv:1708.04896*, 2017. 4
- [46] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818. 7

APPENDIX

VI. PROOF OF SYMMETRIC

Given two independent random variable X and Y , the distribution of product random variable Z , where $Z = XY$, can be found as follows,

$$f_Z(z) = \int_{-\infty}^{\infty} \frac{1}{|t|} f_X(t) f_Y\left(\frac{z}{t}\right) dt. \quad (11)$$

If the distribution of X is continuous at 0, then we have,

$$\begin{aligned} P(Z \leq z) &= P(XY \leq z) \\ &= P(Y \leq \frac{z}{X} | X > 0) P(X > 0) + P(Y \geq \frac{z}{X} | X < 0) \\ &= \int_0^{\infty} P(Y \leq \frac{z}{t}) f_X(t) dt + \int_{-\infty}^0 P(Y \geq \frac{z}{t}) f_X(t) dt \end{aligned} \quad (12)$$

We take the derivation of both sides w.r.t. z and we get,

$$\begin{aligned} f_Z(z) &= \int_0^{\infty} \frac{1}{t} f_Y\left(\frac{z}{t}\right) f_X(t) dt + \int_{-\infty}^0 \frac{-1}{t} f_Y\left(\frac{z}{t}\right) f_X(t) dt \\ &= \int_{-\infty}^{\infty} \frac{1}{|t|} f_X(t) f_Y\left(\frac{z}{t}\right) dt. \end{aligned} \quad (13)$$

Suppose the distribution of random variable Y is symmetric around zero, where $f_Y(y) = f_Y(-y)$. Then we can have,

$$\begin{aligned} f_Z(z) &= \int_{-\infty}^{\infty} \frac{1}{|t|} f_X(t) f_Y\left(\frac{z}{t}\right) dt \\ &= \int_{-\infty}^{\infty} \frac{1}{|t|} f_X(t) f_Y\left(\frac{-z}{t}\right) dt \\ &= f_Z(-z). \end{aligned} \quad (14)$$

Therefore, the distribution of product random variable Z will be symmetric around zero, if the distribution of one of independent random variable X and Y is symmetric around zero.

VII. EXPLANATION OF WA EQUATION FORM

In section III-B, we give out the expression for WA as Eq. (10). Compared with BN, WA does not have the β term since it manipulates the weight directly instead of the activations. If we add a β term to Eq. (10), it will result in,

$$\hat{w} = \gamma \frac{w - \mathbb{E}[w]}{\sqrt{n/2 \cdot \text{Var}[w]}} + \beta. \quad (15)$$

When it multiplies with the input activations \mathbf{x} , we will get an extra $\beta\mathbf{x}$, which is similar to the element in residual blocks. This will introduce the drift of activation variance again, which goes against our original intention.

VIII. ADDITIONAL EXPERIMENTS AND VISUALIZATION

A. Scale factor in WeightAlign

We here conduct an ablation study experiments to validate the effect of our scale factor $\sqrt{n/2}$ in Eq. (10). Specially, we compare our scale factor with a 0.2x our scale factor, 2x our scale factor and 4x our scale factor. The experiments are shown in Fig. 6. We find that a similar scale to $\sqrt{n/2}$ will

lead to similar performance. But our scale $\sqrt{n/2}$ has the best performance. Any other scale factors would cause the failure of training. Thus a proper scaled derivation plays an important role to make the training stable.

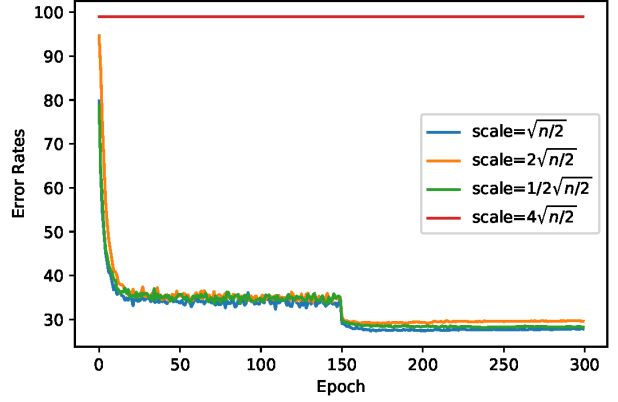


Fig. 6. Ablation study of scale factor. Validation error rates with different scale factors on CIFAR-100 (ResNet18). A slightly different scale factor can cause a failure of training like the red line. A proper scale factor is important to stabilize the optimization. Our scale achieves the best performance among other trainable scale factors.

B. Empirical analysis in Section III-D

We further show activation distributions for different normalization methods over 8 different channels taken from four different layers: the 1st, 3rd, 7th intermediate convolutional layers and the last classification layer. Fig. 7 shows the comparison between baseline and other normalization methods including WA. Fig. 8 shows the cases when IN, LN and GN are used in conjunction with our WA.

C. Visualizations of weights and activations in Section IV-C

We visualize the distributions of weights and activations as epoch increases during training. Fig. 9 and Fig. 10 show weights distributions of a single channel in a convolutional layer. The weights distributions of WA and BN are smooth and symmetric around zero, while the ones of other normalization methods are rough or asymmetric. Adding WA with LN, IN and GN smooths and symmetrizes the weights distributions of channels. Fig. 11 and Fig. 12 show activation distributions of a single channel in a convolutional layer. Similarly, the activation distributions of WA and BN are smooth and symmetric around zero, while the ones of other normalization methods are rough or asymmetric. Adding WA with LN, IN and GN smooths and symmetrizes the activation distributions of channels.

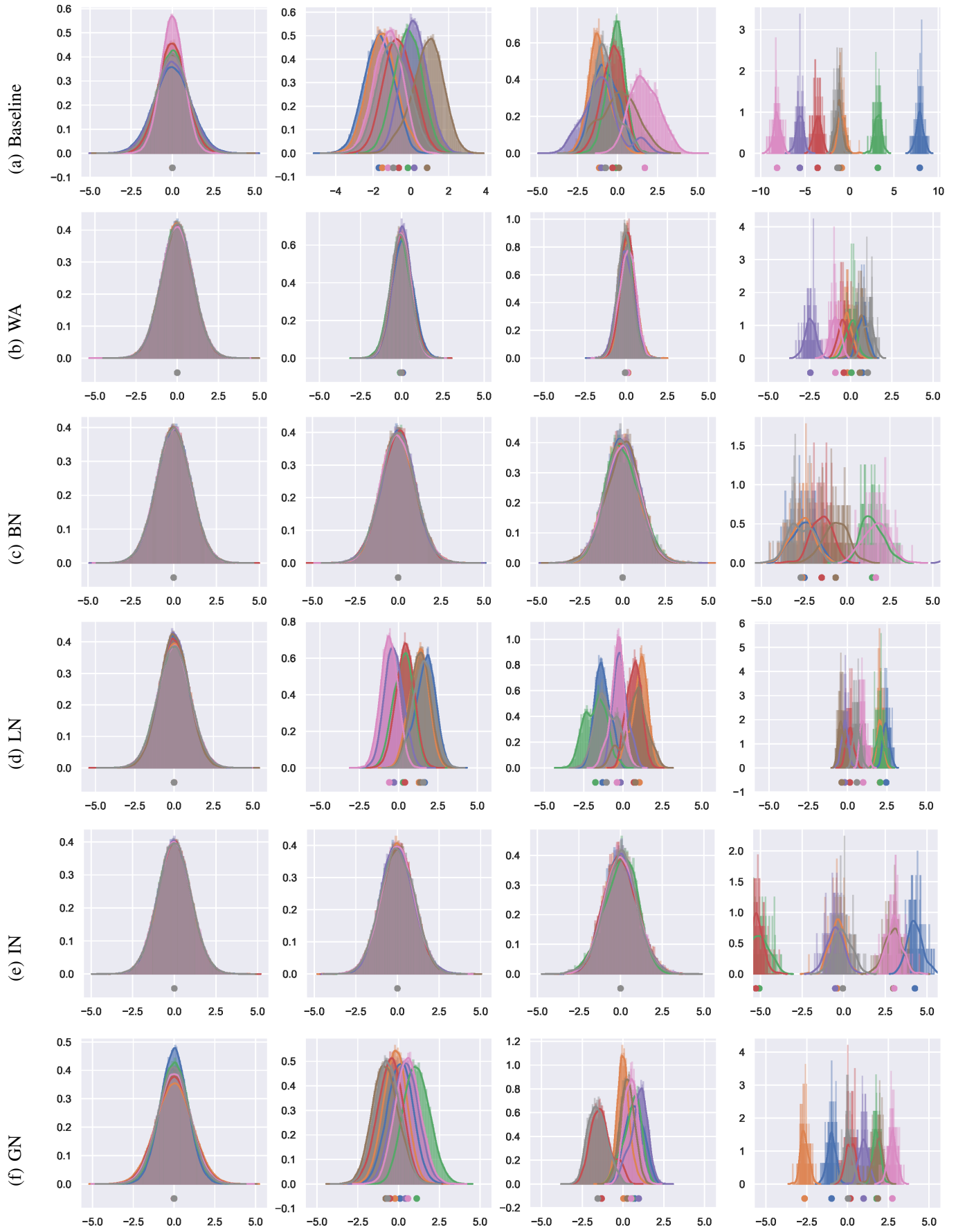


Fig. 7. Each color represents the activation distribution of different channels for different layers. The first three columns denote 1st, 3rd, 7th convolutional layers and the last one presents the last classification layer before softmax. All normalization methods can reduce internal covariate shift to some extent comparing with baseline.

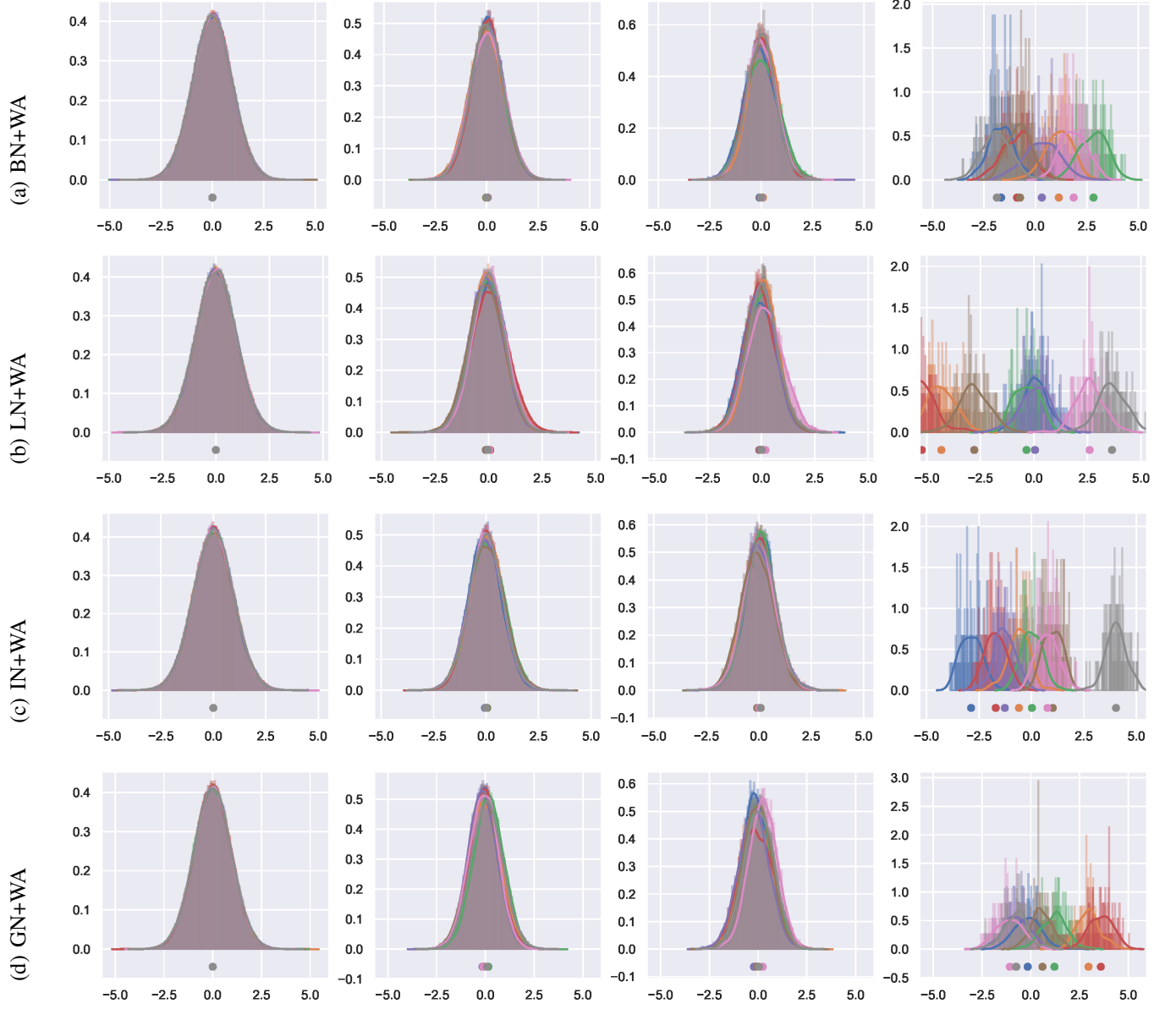


Fig. 8. The proposed WA can be used in conjunction with BN, LN, IN and GN. The first three columns denote 1st, 3rd, 7th convolutional layers and the last one presents the last classification layer before softmax. Note that the activation we plot here is before passing through specific normalization layer.

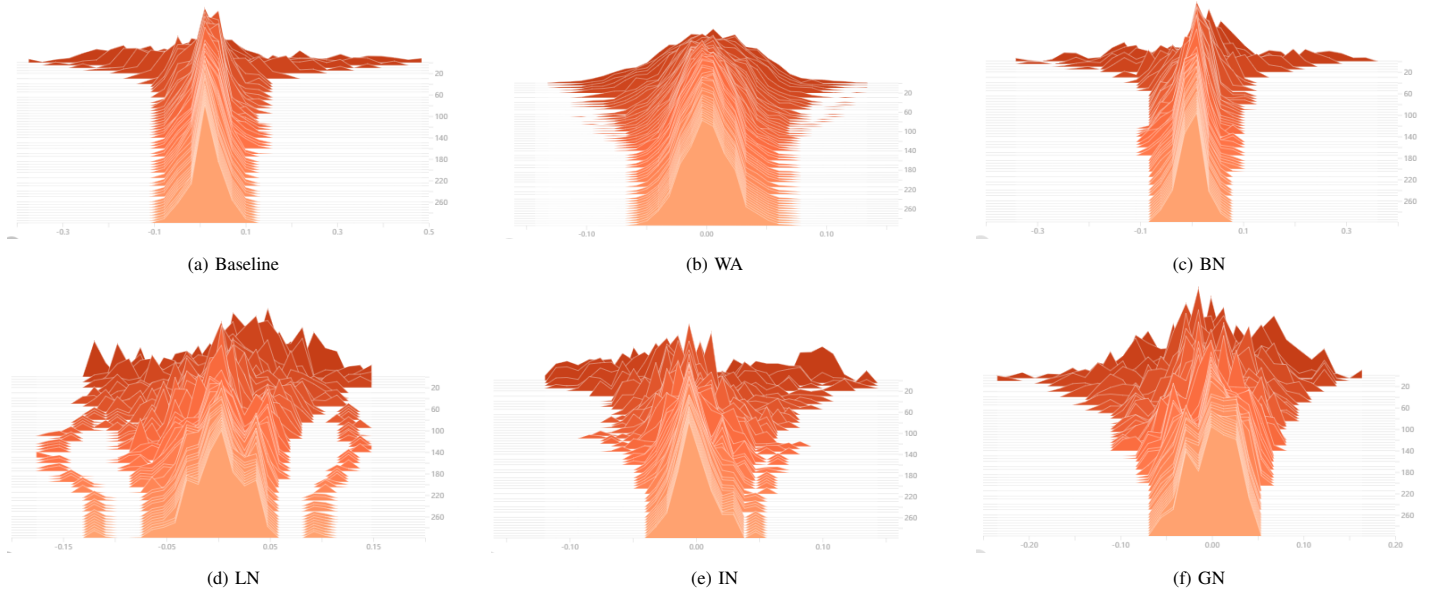


Fig. 9. Weights distributions of a single channel in a convolutional layer for different normalization methods. WA and BN have smooth and symmetric weight distributions.

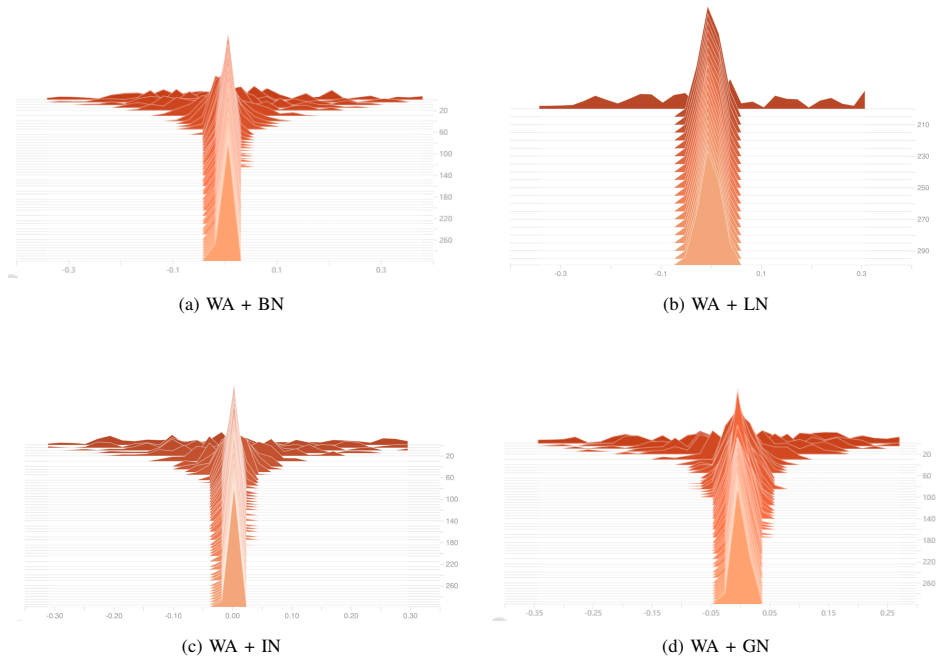


Fig. 10. Weights distributions of a single channel in a convolutional layer for different normalization methods in conjunction with WA method. Adding WA smooths and symmetrizes the weight distributions.

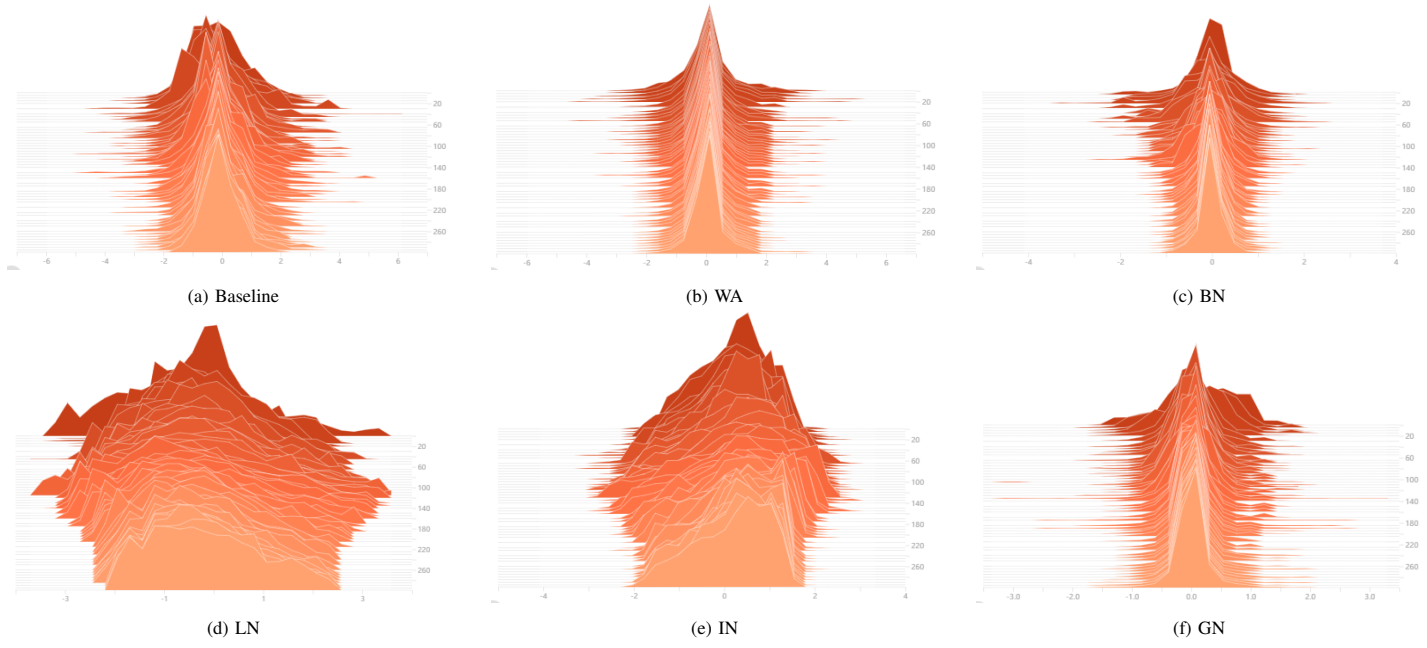


Fig. 11. Activation distributions of a single channel in a convolutional layer for different normalization methods. WA and BN have smooth and symmetric weight distributions.

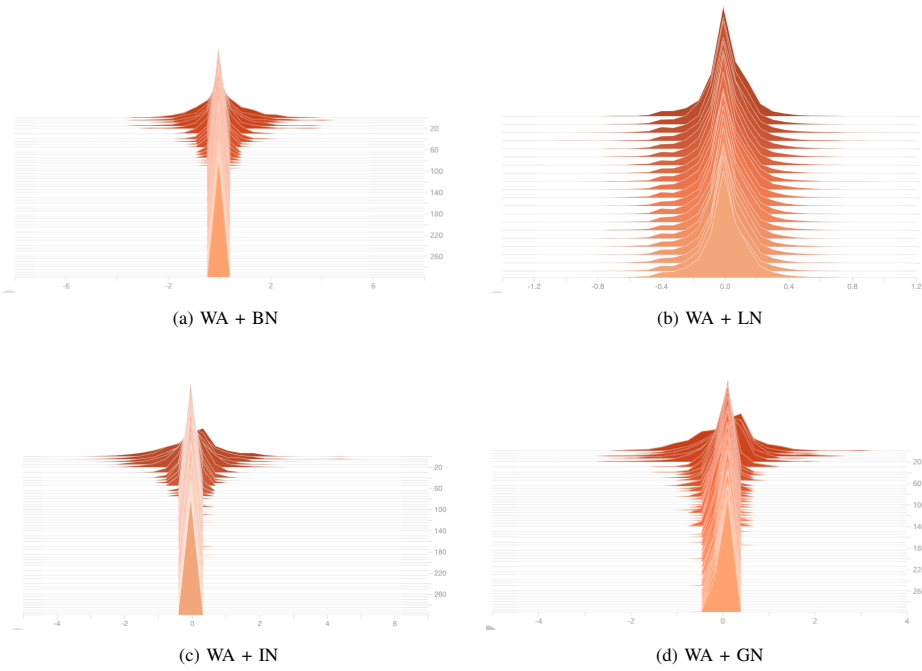


Fig. 12. Activation distributions of a single channel in a convolutional layer for different normalization methods in conjunction with WA method. Adding WA smooths and symmetrizes the activation distributions.