ASSOCIATE NORMALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Normalization is a key technique to improve the training of deep neural networks. However, the existing normalization methods often merely rely on the back-propagation processes to learn the rescaling parameters. That is, the existing methods treat all *different* input features under the *same* distribution, which may limit the feature expressiveness of the normalization module. We present Associate Normalization (AssocNorm) to overcome the above limitation. AssocNorm extracts the useful information from input features and associates them with rescaling parameters predicted by an auto-encoder-like neural network. Therefore, AssocNorm learns the rescaling parameters via both the back-propagation and the association with input features. Furthermore, AssocNorm normalizes the features of each example individually, so the accuracy is relatively stable for different batch sizes. The experimental results show that AssocNorm outperforms the existing normalization methods on several benchmark datasets under various hyper-parameter settings.

1 INTRODUCTION

The technique of *normalization* plays a key role to make gradient propagation more stable during training a deep network. In practice, a normalization mechanism aims to normalize the output of a given layer such that the vanishing gradient problem can be suppressed and hence to reduce the oscillations in output distribution. In this way, speeding up the training process and improving the generalization capability of the trained deep networks are available.

A normalization mechanism usually contains two stages: *standardization* and *rescaling*. The standardization stage aims to regularize the input feature map $\mathbf{x}$ with its mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\gamma}$ by

$$\mathbf{x}_s = \frac{\mathbf{x} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\gamma} + \epsilon}}\,, \tag{1}$$

where $\mathbf{x}_s$ is a standardized input feature map. At the rescaling stage, the standardized feature map $\mathbf{x}_s$ is rescaled with the learned weight $\boldsymbol{\omega}$ and bias $\boldsymbol{\beta}$. The rescaling is used to map $\mathbf{x}_s$ into another feature space by

$$\mathbf{x}_n = \boldsymbol{\omega} \times \mathbf{x}_s + \boldsymbol{\beta}\,, \tag{2}$$

where $\mathbf{x}_n$ is the final output of the entire normalization process. Notice that the learned weight $\boldsymbol{\omega}$ and bias $\boldsymbol{\beta}$ are used to stand as the roles of mean and variance in Eq. (2). The two-stage normalization process is beneficial to the training of deep neural networks by making the network trainable and the training process more stable.

The existing normalization mechanisms mainly focus on studying the *standardization stage* to tackle the issues under various circumstances. In contrast, as far as we know, the *rescaling stage* is less investigated and its related improvements remain to be explored. We observe that existing mechanisms for estimating the rescaling parameters often merely rely on the back-propagation process without considering the association between the standardization stage and the rescaling stage. As a result, the low association-quality causes information loss while data flow is passing these two stages. We argue that the lack of this kind of association may lead to a performance bottleneck in the normalization mechanism.

The proposed Associate Normalization (AssocNorm) aims to keep the association-quality across the standardization stage and the rescaling stage. The design of AssocNorm is inspired by residual networks (He et al., 2016), which use the previously visited feature map for guiding the learning of

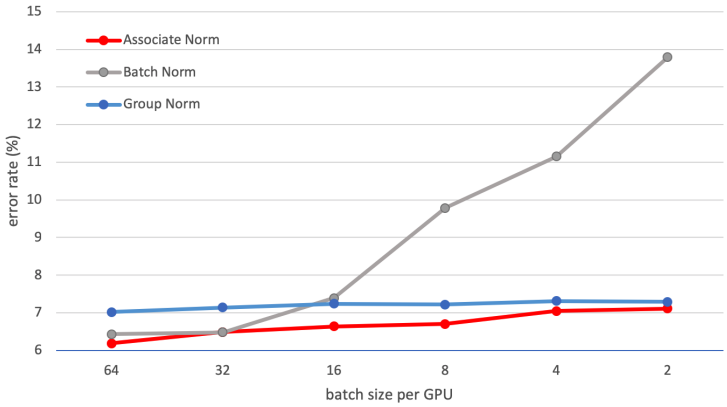

Figure 1: **CIFAR-10 classification error rate against the batch size per GPU.** The evaluation model is ResNet-101. The result shows that the AssocNorm has the best performance on error rates compared with the batch normalization and the group normalization.

the current feature map. Namely, the learning of weight ω and bias β in AssocNorm is following the clue from the input feature map in the standardization stage rather than merely relying on the back-propagation as previous methods do. In practice, we set a shortcut between the input feature map $\mathbf{x}$ with the weight ω and the bias β in the rescaling stage.

AssocNorm normalizes the features within each example individually. This scheme is analogous to Group Normalization (Wu & He, 2018), which performs the normalization without using the batch dimension. The advantage of this scheme is that, since the normalization is not related to the batch size, the performance of AssocNorm more robust to various batch sizes.

An overview of the proposed AssocNorm normalization mechanism is shown in Figure 2. The contributions of this work are summarized as follows:

1. AssocNorm provides a novel way to associate the rescaling parameters with the input feature maps rather than deciding the parameters merely from back-propagation.
2. The number of variables in AssocNorm is small and does not add too much computation burden. For ResNet, the total number of variables would only increase at most 0.05% if using AssocNorm.
3. The experimental results show that AssocNorm performs stably well under various batch sizes.
4. We conduct extensive experiments on several datasets to analyze and compare the properties of AssocNorm with other normalization mechanisms.

It is worthwhile to mention that AssocNorm is comparable or even surpasses Batch Normalization in terms of validation error rates. Note that the performances of most example-independent normalization methods, such as Instance Normalization, Layer Normalization, and Group Normalization, are in general not as good as Batch Normalization when a large batch size is used.

2 Related Work

2.1 Normalization in Deep Neural Network

Gradient-based learning may suffer from the well-known problems of exploding gradient or vanishing gradient, and it has been demonstrated that a normalization mechanism is an effective way to control the degree of such problems.

Several well-known normalization mechanisms have been proposed with the development of deep neural networks. AlexNet (Krizhevsky et al., 2012) and its follow-up models (Sermanet et al., 2014; Szegedy et al., 2015) adopt Local Response Normalization (LRN) (Lyu & Simoncelli, 2008;

Jarrett et al., 2009) to compute the mean and variance of the same spatial locations across several neighboring feature maps for standardizing to the middle one. However, this kind of normalization only focuses on the statistics in a small neighborhood per pixel.

As suggested by its name, Batch Normalization (BN) (Ioffe & Szegedy, 2015) provides a batch-wise normalization method that respectively centers and scales by mean and variance across the whole mini-batch and then rescales the result. Decorrelated Batch Normalization (Huang et al., 2018b) improves Batch Normalization by adding an extra whitening procedure at the standardization stage. For batch-wise normalization mechanisms, the calculation of mean and variance relies on the whole batch. The effectiveness of normalization may degrade when the batch size is not sufficient to support the statistic calculation. To ease the issue of degradation, Batch Renormalization (Ioffe, 2017) suggests adding more learnable parameters in BN.

Several normalization methods (Arpit et al., 2016; Ba et al., 2016; Ulyanov et al., 2016; Ren et al., 2017; Wu & He, 2018) inherit the notion of the Batch Normalization but focus on the manipulations at the standardize stage. Layer Normalization (LN) (Ba et al., 2016) operates along the channel dimension and standardizes the features from one single batch by the batch's own mean and variance. Instance Normalization (IN) (Ulyanov et al., 2016) standardizes each feature map with respect to each sample. Group Normalization (GN) (Wu & He, 2018) divides the feature channels within each batch into several groups and then performs the standardization for each group.

Another way to do normalization is operating on the filter weights instead of operating on the feature maps. For example, Weight Normalization (Salimans & Kingma, 2016) and Orthogonal Weight Normalization (Huang et al., 2018a) present this kind of normalization to address some recognition tasks.

In sum, we observe that the existing normalization methods merely focus on manipulating the learning of parameters at the *standardization stage*. Without considering the association between the standardization stage and the rescaling stage, the parameters learned for rescaling may have a low correlation to the parameters for standardization.

2.2 Style Transfer with Rescaling Parameters

The goal of a style transfer task is to transfer arbitrary visual styles from one image or video to another. Likewise, domain adaption aims to enable a function learned from one domain to work as well in another domain. One solution to this kind of task is manipulating the learned rescaling parameters.

Adaptive Instance Normalization (Huang & Belongie, 2017) applies the rescaling parameters generated by another domain to the feature maps of the current domain via Instance Normalization. Dynamic Layer Normalization (Kim et al., 2017) generates the rescaling parameters by different speakers and environments for adaptive neural acoustic modeling via Layer Normalization.

The core idea of using the learned rescaling parameters to address the tasks of style transfer or domain adaption is similar to the normalization process. The original distribution of one domain is standardized and then mapped to the target distribution in the target domain. Hence, the rescaling parameters learned from the target distribution can be used to recover the original distribution in the target domain.

3 Associate Normalization

This section describes the proposed two-stage normalization mechanism: *Associate Normalization* (AssocNorm). Figure 2 shows an overview of AssocNorm. The first stage is *standardization*, which regularizes the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\gamma}$ of the input feature map $\mathbf{x}$ for standardizing the distribution of the feature map. The second stage is *rescaling*, which rescales the standardized feature map $\mathbf{x}_s$ for recovering the representation ability of the feature map $\mathbf{x}$. The rescaling stage uses an auto-encoder to predict the rescaling parameters, *i.e.*, weight $\boldsymbol{\omega}$ and bias $\boldsymbol{\beta}$, with respect to the input feature map $\mathbf{x}$ instead of generating the rescaling parameter without reasonable grounding.

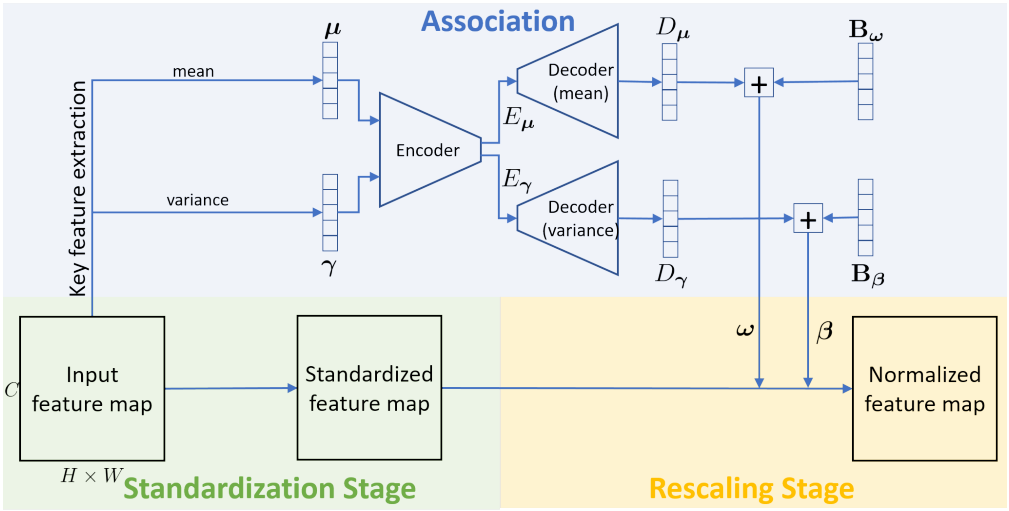

Figure 2: An overview of the proposed AssocNorm normalization mechanism. AssocNorm divides the C channels of the input into groups for computing the mean and variance per group as the key features $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$. An auto-encoder is then used to associate the features $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$ of the input feature map to the rescaling parameters $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$ via its outputs $D_{\boldsymbol{\mu}}$ and $D_{\boldsymbol{\gamma}}$.

3.1 Standardization Stage

The goal of the standardization stage is to regularize the distribution of the input feature map, which is often done by forcing the distribution to have zero mean and unit variance. Existing normalization mechanisms usually focus on designing this stage for fitting various circumstances.

AssocNorm adopts Group Normalization's standardization process. Group Normalization (GN) divides the whole layer into several groups along its channel axis, each group calculates its own mean and variance for standardization. The reason we choose GN is that it not only has strong ability on several visual tasks, but, moreover, maintains consistency with the way to extract key features from input feature maps, which will be discussed in Section 3.3.1.

3.2 Rescaling Stage

The goal of the rescaling stage is to recover the representation ability of the input feature map. The parameters for recovering the representation ability are usually merely learned via the back-propagation process. In contrast, AssocNorm suggests to predict the parameters with additional association between the standardization stage and the rescaling stage. In the following, we detail the process of extracting the key features of the input feature map $\mathbf{x}$. The auto-encoder for predicting the rescaling parameters will be presented in Section 3.3.

3.3 Stage Association within Normalization

For keeping the association-quality between the standardization stage and the rescaling stage with controlled computational cost, we implement AssocNorm with *auto-encoder* networks. An overview of the components is shown in Figure 2. An auto-encoder is used to predict the rescaling parameters $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$ concerning the pre-computed mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\gamma}$ of the input feature map $\mathbf{x}$. In comparison with the existing methods that merely learn the parameters $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$ via back-propagation, our experiments show that the parameter-learning characteristic with the additional consideration of $\boldsymbol{\omega}$ and the $\boldsymbol{\beta}$ is helpful.

3.3.1 KEY FEATURE EXTRACTION

AssocNorm uses an auto-encoder to predict the weights $\boldsymbol{\omega}$ and bias $\boldsymbol{\beta}$ as the rescaling parameters for recovering the distribution of the feature map $\mathbf{x}$. We have observed that directly encoding the entire input feature map would degrade the prediction accuracy, which might be due to overfitting. Instead of using the entire feature map $\mathbf{x}$ as the input for the auto-encoder, we suggest using the mean μ and variance γ of $\mathbf{x}$ for characterizing it, since the mean and variance not only represent some global statistics of input feature maps, but also share the similar attributes with the weights $\boldsymbol{\omega}$ and bias $\boldsymbol{\beta}$ we seek to generate. Here we define the key features (characteristic features) as the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\gamma}$ extracted from the input feature map $\mathbf{x}$. The experimental results demonstrate that, AssocNorm, which uses a compact auto-encoder, is able to predict $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$ well for recovering the distribution of $\mathbf{x}$.

Furthermore, for higher performance and lower calculation burden, we extract the key features from each group of input feature maps rather than a single feature map. For a specific layer comprising C channels as feature maps $f_1, f_2, \ldots, f_C$, we evenly partition these feature maps into N groups $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_N$. The mean and variance of the whole layer are hence denoted as a vector of length N, namely $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_N]$ and $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_N]$. AssocNorm computes the mean μ_n and variance γ_n for a given feature-map group $\mathbf{f}_n$ as

$$\begin{cases} \mu_n = \frac{1}{H \times W \times C/N} \sum_{f \in \mathbf{f}_n} \sum_{i=1}^{H} \sum_{j=1}^{W} f^{i,j}\,, \\ \gamma_n = \frac{1}{H \times W \times C/N} \sum_{f \in \mathbf{f}_n} \sum_{i=1}^{H} \sum_{j=1}^{W} (f^{i,j} - \mu_n)^2\,, \end{cases} \tag{3}$$

where C/N is the number of feature maps in a group, f denotes a feature map of group $\mathbf{f}_n$. Further discussions for key feature extraction can be found in Section 4.3.

3.3.2 ENCODER

The goal of the encoder in AssocNorm is to summarize the information of an input feature map $\mathbf{x}$'s key features through an embedding. Besides, we expect the subsequent rescaling parameters can be jointly learned from the same embedded information.

In our implementation, the encoder comprises one *fully connected layer* and one *activation function*. The fully connected layer can model not only the individual elements of the key features but also the correlations between elements. Using an activation function allows us to extract non-linear information. The embedded vectors, which encode the mean and the variance of the grouped input feature maps, are obtained by

$$\begin{cases} E_{\boldsymbol{\mu}} = \mathrm{ReLU}(\mathbf{W}_1 \boldsymbol{\mu})\,, \\ E_{\boldsymbol{\gamma}} = \mathrm{ReLU}(\mathbf{W}_1 \boldsymbol{\gamma})\,, \end{cases} \tag{4}$$

where $E_{\boldsymbol{\mu}}$ and $E_{\boldsymbol{\gamma}}$ respectively denote the embedded vectors of $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$, $\mathrm{ReLU}(\cdot)$ represents the activation function, and the encoding matrix $\mathbf{W}_1 \in \mathbb{R}^{M \times N}$ with the embedded vector of length M and key feature vectors of length N.

3.3.3 DECODER

The decoder in AssocNorm aims to decode the embedded vectors $E_{\boldsymbol{\mu}}$ and $E_{\boldsymbol{\gamma}}$ into $D_{\boldsymbol{\mu}}$ and $D_{\boldsymbol{\gamma}}$ respectively. $D_{\boldsymbol{\mu}}$ and $D_{\boldsymbol{\gamma}}$ are treated as the contributions of original feature maps to the rescaling parameters $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$.

In our implementation, we use two different *fully connected layers* and two *activation functions*. Using a fully connected layer for decoding is able to summarize the information-rich embedded vectors for predicting the rescaling parameters. Accompanying the decoded vector with an activation function translates the vector values into a suitable range. The decoded vectors, which decode the mean and the variance of the embedded vector are obtained as

$$\begin{cases} D_{\boldsymbol{\mu}} = \mathrm{sigmoid}(\mathbf{W}_2 \mathbf{E}_{\boldsymbol{\mu}})\,, \\ D_{\boldsymbol{\gamma}} = \tanh(\mathbf{W}_3 \mathbf{E}_{\boldsymbol{\gamma}})\,, \end{cases} \tag{5}$$

where both $\mathrm{sigmoid}(\cdot)$ and $\tanh(\cdot)$ represent the activation functions, and the decoding matrices $\mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{N \times M}$.

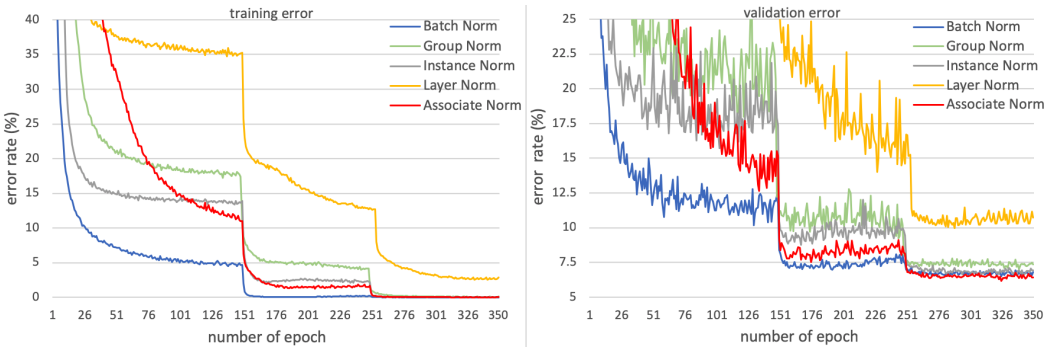

Figure 3: **Comparison of error rate (%) against the number of training epochs using the batch size of** 64**.** Both the training error and the validation error of CIFAR-10 are shown.

CIFAR-10: Error Rate (%)	Methods				
	BN	AssocNorm	GN	IN	LN
value	6.43	**6.19**	7.02	6.54	9.98
vs. BN	-	**-0.24**	+0.59	+0.11	+3.55

Table 1: **Comparison of the error rate (%) in CIFAR-10 using** 350 **training epochs.**

3.3.4 BIAS

Notice that each decoded vector $D_{\boldsymbol{\mu}}$ or $D_{\boldsymbol{\gamma}}$ predicted from the auto-decoder needs to align with a corresponding bias value. The bias term plays a similar role to the rescaling parameters in traditional normalization module. It reflects the desirable distribution of the following layers, since its value is independent with input feature and can only be learned by back-propagation, which makes the information transfer more efficient. Therefore, the bias term is learned to align the decoded vectors as

$$\begin{cases} \boldsymbol{\omega} = D_{\boldsymbol{\mu}} + \mathbf{B}_{\boldsymbol{\omega}} \,, \\ \boldsymbol{\beta} = D_{\boldsymbol{\gamma}} + \mathbf{B}_{\boldsymbol{\beta}} \,, \end{cases} \tag{6}$$

where $\mathbf{B}_{\boldsymbol{\omega}}$ and $\mathbf{B}_{\boldsymbol{\beta}}$ denote the learned bias vector of $\boldsymbol{\omega}$ and $\boldsymbol{\beta}$, respectively.

4 EXPERIMENTS

We evaluate the proposed AssocNorm on CIFAR-10 and CIFAR-100 datasets. The AssocNorm is compared with various state-of-the-art normalization methods for training deep neural networks, including Batch Normalization (BN), Layer Normalization (LN), Instance Normalization (IN), and Group Normalization (GN).

Implementation Details. AssocNorm built upon the ResNet-101 model (He et al., 2016) for evaluating all experiments. We initial all parameters with standard normal distribution yet we initial all rescaling weight $\boldsymbol{\omega}$ by 1 and bias $\boldsymbol{\beta}$ by 0. Unless otherwise stated, both GN and AssocNorm set the number of groups equal to 32, batch size of all normalization methods euqal to 64 and we use only 1 GPU for training. We use SGD as the optimizer with momentum 0.9 and weight decay 0.0005. Each normalization methods are trained with 350 epochs. The learning rate is initialized with 0.1 and decreased by 0.1 at 150-th and 250-th epoch.

4.1 IMAGE CLASSIFICATION WITH LARGE BATCH SIZE

4.1.1 CIFAR-10

In this section, we compare all normalization methods on ResNet-101 in CIFAR-10 dataset. The CIFAR-10 dataset (Krizhevsky & Hinton, 2009; Torralba et al., 2008) contains 10 different classes for the image classification task. Each class comprises 5,000 training images and 1,000 testing images. The results are shown in Figure 3 and Table 1.

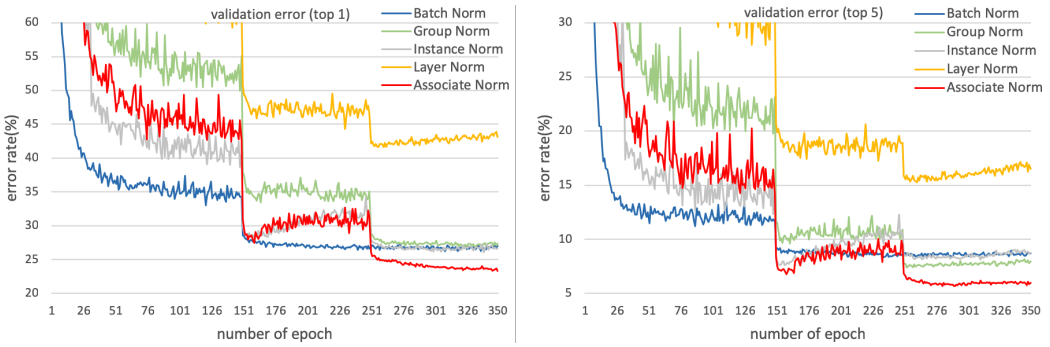

Figure 4: **Comparison of error rate (%) against the number of training epochs using the batch size of** 64**.** Both the top-1 error rate and the top-5 error rate of CIFAR-100 are shown .

Top-1 Error Rate (%)	Methods				
	BN	AssocNorm	GN	IN	LN
value	26.28	**23.31**	26.94	26.08	41.61
$vs.$ BN	-	**-2.97**	+0.66	-0.20	+15.33
Top-5 Error Rate (%)	**Methods**				
	BN	AssocNorm	GN	IN	LN
value	9.37	**5.71**	7.02	7.60	15.26
$vs.$ BN	-	**-3.66**	-2.35	-1.77	+5.89

Table 2: **Comparison of the error rates (%) in CIFAR-100 using** 350 **training epochs.** The top portion shows the top-1 error rate and the bottom portion shows the top-5 error rate.

Figure 3 shows the comparison of error rate against the number of training epochs using the batch size of 64. In Fig. 3, AssocNorm outperforms than Group Normalization, Instance Normalization, and Layer Normalization after training more than 100 epochs. Though the Batch Normalization, which obviously prefers the large batch size as shown in Table 3, shows the outstanding performance, AssocNorm surpass it eventually. Table 1 shows the error rate with 350 training epochs. Notice that we show the relative difference w.r.t. BN in the last row of Table 1, and only AssocNorm can surpass its performance. It is worthwhile to mention that no existing state-of-the-art normalization methods can surpass the BN using a large batch size to reach such a low error rate in the CIFAR-10 classification task to the best of our knowledge.

4.1.2 CIFAR-100

In this section, we compare all normalization methods on ResNet-101 in the CIFAR-100 dataset (Krizhevsky & Hinton, 2009; Torralba et al., 2008), which contains 100 different classes for the image classification task. Each class comprises 500 training images and 100 testing images. We train AssocNorm on the training set of 50,000 images and evaluate on the validation set of 10,000 images. The results are shown in Figure 4 and Table 2.

Figure 4 shows the comparison of error rate against the number of training epochs. In Figure 4, AssocNorm outperforms than Group Normalization, Instance Normalization, and Layer Normalization after training more than 250 epochs. Table 2 shows the error rate with 350 training epochs. We provide the relative difference w.r.t. BN in the last row of the both two portions in Table 2. Our method surpasses the first runner-up in 2.77% (IN) and 1.31% (GN) on the top-1 and top-5 error rate metrics, respectively.

In sum, the experiments of classification task in both CIFAR-10 and CIFAR-100 datasets demonstrate that AssocNorm outperforms the state-of-the-art methods on the error rate. The performance of the proposed normalization is evident.

CIFAR-10: Error Rate (%)		Batch Size					
		64	32	16	8	4	2
Methods	AssocNorm	**6.19**	6.49	**6.64**	**6.70**	**7.05**	**7.11**
	GN	7.02	7.14	7.24	7.22	7.31	7.29
	BN	6.43	**6.48**	7.39	9.78	11.15	13.79
	GN $vs.$ AssocNorm	+0.83	+0.65	+0.60	+0.52	+0.26	+0.18
	BN $vs.$ AssocNorm	+0.24	-0.01	+0.75	+3.08	+4.10	+6.68

Table 3: **Comparison of the error rate (%) against the number of batch sizes.**

Increased-Parameter Ratio	Model				
	ResNet-18	ResNet-34	ResNet-50	ResNet-101	ResNet-152
Group Normalization	0.03 %	0.03 %	0.04 %	0.04 %	0.05 %
Instance Normalization	2.79 %	2.46%	20.70%	20.31%	20.18%

Table 4: **The number of increased parameters concerning the different key feature extraction strategies.**

4.2 Image Classification with Various Batch Sizes

We conduct the experiment to compare the error rate against the various batch sizes for comparing AssocNorm to the state-of-the-art normalization methods in CIFAR-10. The experiment considers batch sizes of $\{64, 32, 16, 8, 4, 2\}$ per GPU without changing the other hyper-parameters. The results are shown in Figure 1 and Table 3.

Figure 1 clearly shows that both AssocNorm and GN are not sensitive to the batch size. In contrast, the BN obviously prefer a larger batch size. Table. 3 shows that the proposed AssocNorm has the lowest error rates among all batch sizes except the batch size of 32, which is merely 0.01% worse than BN. On average, AssocNorm has a lower error rate than GN in 0.51% and lower error rate than BN in 2.47% among the testing batch sizes.

Discussion. Table 3 shows that AssocNorm outperform than GN among all batch sizes. Since we constraint all hyper-parameters of AssocNorm the same to BN, we think the performance gain derived from the association, which is built upon our auto-encoder, between the standardization stage and rescaling stage. As a result, while a normalization task learning the parameters of the rescaling stage, it is helpful to leverage both the cross-stage association and the back-propagation process.

In sum, the experiments demonstrate that AssocNorm has better performance than the state-of-the-art methods on error rate and batch-size robustness. With the large batch size, AssocNorm can even surpass the BN, which prefers a large batch size, to be the new state-of-the-art in the CIFAR-10 classification task.

4.3 Alternative Key Feature Extraction

For representing key features, AssocNorm divides the input channels into groups for computing the mean and variance per group. As mentioned in Section 2.1, the existing normalization methods, such as Batch Normalization (BN), Layer Normalization (LN), Instance Normalization (IN), and Group Normalization (GN), have their own way to extract the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\gamma}$ from input $\mathbf{x}$. To make our normalization mechanism robust to various batch sizes, the way in BN is out of our consideration. Moreover, we also ignore using the LN's method, since LN merely generates one pair of mean and variance for all feature maps in a layer, such few data is not available for training our auto-encoder-like network. Therefore, our key feature extraction strategy considers the implementation as IN or GN. Figure 5 shows the comparison of error rate against the number of training epochs using the key feature extraction strategy as GN or IN. Table 4 provides the number of increased parameters concerning the different key feature extraction strategies.

Figure 5 shows that the performance using key feature extraction as the Group Normalization is usually better than that using the Instance Normalization. From the perspective of the increased

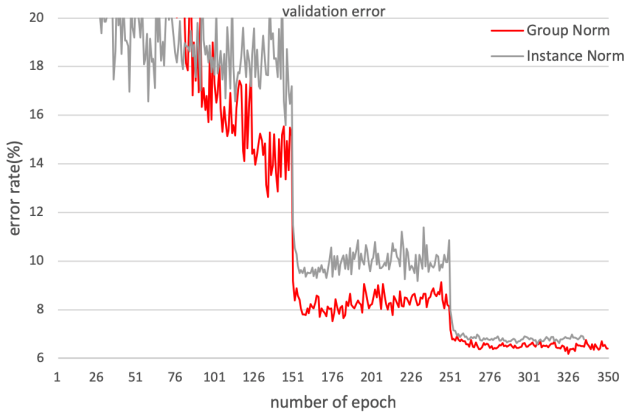

Figure 5: **The error rate (%) against the number of training epochs concerning the different key feature extraction strategies.** This figure shows the CIFAR-10 validation error.

Error Rate (%)	Methods			
	$a(\mathbf{WE})$	$a(\mathbf{WE}+\mathbf{B})$	$a(\mathbf{WE})\times\mathbf{B}$	$a(\mathbf{WE})+\mathbf{B}$
value	7.04	6.46	6.49	**6.11**
$vs.$ $a(\mathbf{WE})+\mathbf{B}$	+0.94	+0.35	+0.38	-

Table 5: **Comparison of the error rate (%) using the decoded vectors and the bias term.**

number of parameters, Table 4 provides the other information for choosing the strategy of key feature extraction. In Table 4, the increased parameters using key feature extraction as GN is quite small. The fewer parameters of GN is because it partition each input C channels into M groups, hence the increased parameters depend on the small value of M instead of the big value of C. In contrast, the increased parameters of IN depend on the value of C. Therefore, the matrices $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ in the auto-encoder of AssocNorm benefit more from GN.

In sum, the experiments demonstrate that using the key feature extraction strategy as GN obtain not only the lower error rate but also the fewer number of increased parameters.

4.4 Bias term

As early mentioned in Section 3.3.4, the decoded vectors predicted from the auto-decoder has to cooperate with a corresponding bias value for pursuing the better normalization performance. To analysis the impacts of the decoded vectors and the bias term, we conduct the experiment to compare with four combinations as the following settings: *i*) $a(\mathbf{WE})$: ignore the bias term. *ii*) $a(\mathbf{WE}+\mathbf{B})$: add the bias term before activation. *iii*) $a(\mathbf{WE})\times\mathbf{B}$: multiply the bias term. *iv*) $a(\mathbf{WE})+\mathbf{B}$: add the bias term as proposed. Notice that we use $a(\cdot)$ to denote an activation function.

In this experiment, the error rates using 350 training epochs of $a(\mathbf{WE})$, $a(\mathbf{WE}+\mathbf{B})$, $a(\mathbf{WE})\times\mathbf{B}$, and $a(\mathbf{WE})+\mathbf{B}$ are 7.04%, 6.46%, 6.49%, and 6.11%, respectively. The result shows that predicting the rescaling parameters with the aids of the bias term is evident, and the suggested combination, *i.e.*, $a(\mathbf{WE})+\mathbf{B}$, outperforms the others.

5 Conclusion

In this paper, we have present AssocNorm extracting the key features and associating them with rescaling parameters, that predicted by an auto-encoder-like neural network, for normalization task. As a result, AssocNorm provides the way to learn the rescaling parameters via both the back-propagation and the association with input features. The experiments demonstrate that a deep network equips with AssocNorm can improve the performance and robust to various batch sizes with quite a few increased parameters.

REFERENCES

Devansh Arpit, Yingbo Zhou, Bhargava Urala Kota, and Venu Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. In *ICML*, pp. 1168–1176, 2016.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, Yongliang Wang, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *AAAI*, 2018a.

Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *CVPR*, 2018b.

Xun Huang and Serge J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, pp. 1510–1519, 2017.

Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *NIPS*, pp. 1942–1950, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pp. 448–456, 2015.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, pp. 2146–2153, 2009.

Taesup Kim, Inchul Song, and Yoshua Bengio. Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition. In *Interspeech*, pp. 2411–2415, 2017.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1106–1114, 2012.

Siwei Lyu and Eero P. Simoncelli. Nonlinear image representation using divisive normalization. In *CVPR*, 2008.

Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H. Sinz, and Richard S. Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. In *ICLR*, 2017.

Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, pp. 901, 2016.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pp. 1–9, 2015.

Antonio Torralba, Robert Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11): 1958–1970, 2008.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.