Exemplar Normalization for Learning Deep Representation

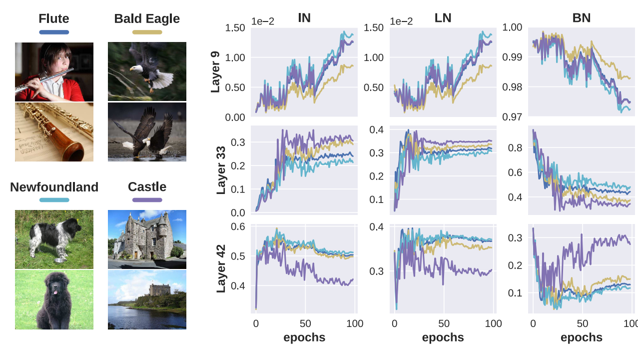
Ruimao Zhang[1]*, Zhanglin Peng[1]*, Lingyun Wu[1], Zhen Li[3,4], Ping Luo[2]
[1] SenseTime Research, [2] The University of Hong Kong,
[3] The Chinese University of Hong Kong (Shenzhen), [4] Shenzhen Research Institute of Big Data
{zhangruimao, pengzhanglin, wulingyun}@sensetime.com, lizhen@cuhk.edu.cn, pluo.lhi@gmail.com

Abstract

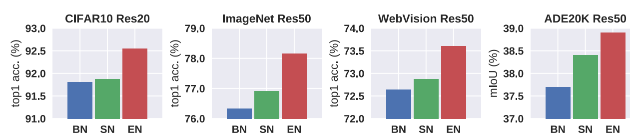
Normalization techniques are important in different advanced neural networks and different tasks. This work investigates a novel dynamic learning-to-normalize (L2N) problem by proposing Exemplar Normalization (EN), which is able to learn different normalization methods for different convolutional layers and image samples of a deep network. EN significantly improves flexibility of the recently proposed switchable normalization (SN), which solves a static L2N problem by linearly combining several normalizers in each normalization layer (the combination is the same for all samples). Instead of directly employing a multi-layer perceptron (MLP) to learn data-dependant parameters as conditional batch normalization (cBN) did, the internal architecture of EN is carefully designed to stabilize its optimization, leading to many appealing benefits. (1) EN enables different convolutional layers, image samples, categories, benchmarks, and tasks to use different normalization methods, shedding light on analyzing them in a holistic view. (2) EN is effective for various network architectures and tasks. (3) It could replace any normalization layers in a deep network and still produce stable model training. Extensive experiments demonstrate the effectiveness of EN in wide spectrum of tasks including image recognition, noisy label learning, and semantic segmentation. For example, by replacing BN in the ordinary ResNet50, improvement produced by EN is 300% more than that of SN on both ImageNet and the noisy WebVision dataset.

1. Introduction

Normalization techniques are one of the most essential components to improve performance and accelerate training of convolutional neural networks (CNNs). Recently, a family of normalization methods is proposed including batch normalization (BN) [14], instance normalization (IN) [36], layer normalization (LN) [1] and group normalization (GN) [39]. As these methods were designed for different tasks, they often normalize feature maps of CNNs from different dimensions.



(a) The learning dynamic of EN ratios of four categories in three layers.



(b) Performance of EN and its counterparts on various CV tasks.

Figure 1. (a) The proposed Exemplar Normalization (EN) enables different categories to learn to select different normalizers in different layers. The four categories of ImageNet (*i.e*. flute, bald eagle, newfoundland and castle) in three layers (*i.e*. bottom, middle and top) of ResNet50 are presented. (b) EN outperforms its counterparts on various computer vision tasks (*i.e*. image classification, noisy-supervised classification and semantic image segmentation) by using different network architectures.

To combine advantages of the above methods, switchable normalization (SN) [23] and its variant [33] were proposed to learn linear combination of normalizers for each convolutional layer in an end-to-end manner. We term this normalization setting as static 'learning-to-normalize'. Despite the successes of these methods, once a CNN is optimized by using them, it employed the same combination ratios of the normalization methods for all image samples in a dataset, incapable to adapt to different instances and thus

*Equal contribution

rendering suboptimal performance.

As shown in Fig. 1, this work studies a new learning problem, that is, dynamic 'learning-to-normalize', by proposing Exemplar Normalization (EN), which is able to learn arbitrary normalizer for different convolutional layers, image samples, categories, datasets, and tasks in an end-to-end way. Unlike previous conditional batch normalization (cBN) that used multi-layer perceptron (MLP) to learn data-dependent parameters in a normalization layer, suffering from over-fitting easily, the internal architecture of EN is carefully designed to learn data-dependent normalization with merely a few parameters, thus stabilizing training and improving generalization capacity of CNNs.

EN has several appealing benefits. (1) It can be treated as an ***explanation tool*** for CNNs. The exemplar-based important ratios in each EN layer provide information to analyze the properties of different samples, classes, and datasets in various tasks. As shown in Fig. 1(a), by training ResNet50 [9] on ImageNet [6], images from different categories would select different normalizers in the same EN layer, leading to superior performance compared to the ordinary network. (2) EN makes ***versatile design*** of the normalization layer possible, as EN is suitable for various benchmarks and tasks. Compared with state-of-the-art counterparts in Fig. 1(b), EN consistently outperforms them on many benchmarks such as ImageNet [6] for image classification, Webvision [18] for noisy label learning, ADE20K [42] and Cityscapes [5] for semantic segmentation. (3) EN is a ***plug and play module.*** It can be inserted into various CNN architectures such as ResNet [9], Inception v2 [35], and ShuffleNet v2 [26], to replace any normalization layer therein and boost their performance.

The **contributions** of this work are three-fold. (1) We present a novel normalization learning setting named dynamic 'learning-to-normalize', by proposing Exemplar Normalization (EN), which learns to select different normalizers in different normalization layers for different image samples. EN is able to normalize image sample in both training and testing stage. (2) EN provides a flexible way to analyze the selected normalizers in different layers, the relationship among distinct samples and their deep representations. (3) As a new building block, we apply EN to various tasks and network architectures. Extensive experiments show that EN outperforms its counterparts in wide spectrum of benchmarks and tasks. For example, by replacing BN in the ordinary ResNet50 [9], improvement produced by EN is 300% more than that of SN on both ImageNet [6] and the noisy WebVision [18] dataset.

2. Related Work

Many normalization techniques are developed to normalize feature representations [14, 1, 36, 39, 23] or weights of filters [12, 32, 27] to accelerate training and boost generation ability of CNNs. Among them, Batch Normalization (BN) [14], Layer Normalization (LN) [1] and Instance Normalization (IN) [36] are most popular methods that compute statistics with respect to channel, layer, and minibatch respectively. The follow-up Position Normalization [17] normalizes the activations at each spatial position independently across the channels. Besides normalizing different dimensions of the feature maps, another branch of work improved the capability of BN to deal with small batch size, including Group Normalization (GN) [39], Batch Renormalization (BRN) [13], Batch Kalman Normalization (BKN) [37] and Stream Normalization (StN) [20].

In recent studies, using the hybrid of multiple normalizers in a single normalization layer has achieved much attention [29, 28, 24, 30, 25]. For example, Pan *et al.* introduced IBN-Net [29] to improve the generalization ability of CNNs by manually designing the mixture strategy of IN and BN. In [28], Nam *et al.* adopted the same scheme in style transfer, where they employed gated function to learn the important ratios of IN and BN. Luo *et al.* further proposed Switchable Normalization (SN) [23, 22] and its sparse version [33] to extend such a scheme to deal with arbitrary number of normalizers. More recently, Dynamic Normalization (DN) [25] was introduced to estimate the computational pattern of statistics for the specific layer. Our work is motivated by this series of studies, but provides a more flexible way to learn normalization for each sample.

The adaptive normalization methods are also related to us. In [31], Conditional Batch Normalization (cBN) was introduced to learn parameters of BN (*i.e.* scale and offset) adaptively as a function of the input features. Attentive Normalization (AN) [19] learns sample-based coefficients to combine feature maps. In [21], Deecke *et al.* proposed Mode Normalization (MN) to detect modes of data on-the-fly and normalize them. However, these methods are incapable to learn various normalizers for different convolutional layers and images as EN did.

The proposed EN also has a connection with learning data-dependent [15] or dynamic weights [41] in convolution and pooling [16]. The subnet for computation of important ratios is also similar to SE-like [11, 2, 38] attention mechanism in form, but they are technically different. First, SE-like models encourage channels to contribute equally to the feature representation [34], while EN learns to select different normalizers in different layers. Second, SE is plugged into different networks by using different schemes. EN could directly replace other normalization layers.

3. Exemplar Normalization (EN)

3.1. Notation and Background

Overview. We introduce normalization in terms of a 4D tensor, which is the input data of a normalization layer in a mini-batch. Let $\boldsymbol{X} \in \mathbb{R}^{N \times C \times H \times W}$ be the input 4D tensor, where N, C, H, W indicate the number of images, number of channels, channel height and width respectively. Here H and W define the spatial size of a single feature map. Let matrix $\boldsymbol{X}_n \in \mathbb{R}^{C \times HW}$ denote the feature maps of n-th image, where $n \in \{1, 2, ..., N\}$. Different normalizers normalize $\boldsymbol{X}_n$ by removing its mean and standard deviation along different dimensions, performing a formulation

$$\widehat{\boldsymbol{X}}_n = \boldsymbol{\gamma} \, \frac{\boldsymbol{X}_n - \boldsymbol{\mu}^k}{\sqrt{(\boldsymbol{\delta}^k)^2 + \epsilon}} + \boldsymbol{\beta} \tag{1}$$

where $\widehat{\boldsymbol{X}}_n$ is the feature maps after normalization. $\boldsymbol{\mu}^k$ and $\boldsymbol{\delta}^k$ are the vectors of mean and standard deviation calculated by the k-th normalizer. Here we define $k \in$ {BN, IN, LN, GN,...}. The scale parameter $\boldsymbol{\gamma} \in \mathbb{R}^C$ and bias parameter $\boldsymbol{\beta} \in \mathbb{R}^C$ are adopted to re-scale and re-shift the normalized feature maps. ϵ is a small constant to prevent dividing by zero, and both $\sqrt{\cdot}$ and $(\cdot)^2$ are channel-wise operators.

Switchable Normalization (SN). Unlike previous methods that estimated statistics over different dimensions of the input tensor, SN [23, 24] learns a linear combination of statistics of existing normalizers,

$$\widehat{\boldsymbol{X}}_n = \boldsymbol{\gamma} \, \frac{\boldsymbol{X}_n - \sum_k \lambda^k \boldsymbol{\mu}^k}{\sqrt{\sum_k \lambda^k \, (\boldsymbol{\delta}^k)^2 + \epsilon}} + \boldsymbol{\beta} \tag{2}$$

where $\lambda^k \in [0, 1]$ is a learnable parameter corresponding to the k-th normalizer, and $\sum_k \lambda^k = 1$. In practice, this important ratio is calculated by using the softmax function. The important ratios for mean and variance can be also different. Although SN [23] outperforms the individual normalizer in various tasks, it solves a static 'learning-to-normalize' problem by switching among several normalizers in each layer. Once SN is learned, its important ratios are fixed for the entire dataset. Thus the flexibility of SN is limited and it suffers from the bias between the training and the test set, leading to sub-optimal results.

In this paper, Exemplar Normalization (EN) is proposed to investigate a dynamic 'learning-to-normalize' problem, which learns different data-dependant normalizations for different image samples in each layer. EN extremely expands the flexibility of SN, while retaining SN's advantages of differential learning, stability of model training, and capability in multiple tasks.

3.2. Formulation of EN

Given input feature maps $\boldsymbol{X}_n$, Exemplar Normalization (EN) is defined by

$$\widehat{\boldsymbol{X}}_n = \sum_k \; \boldsymbol{\gamma}^k (\, \lambda_n^k \, \frac{\boldsymbol{X}_n - \boldsymbol{\mu}^k}{\sqrt{(\boldsymbol{\delta}^k)^2 + \epsilon}} \,) + \boldsymbol{\beta}^k \tag{3}$$

where $\lambda_n^k \in [0, 1]$ indicates the important ratio of the k-th normalizer for the n-th sample. Similar with SN, we

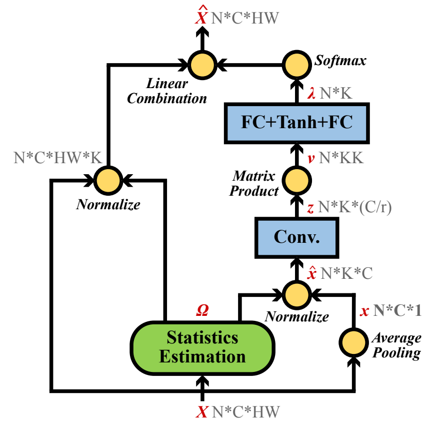


Figure 2. Illustration of the Exemplar Normalization (EN) layer, which is able to learn the sample-based important ratios to normalize the input feature maps by using multiple normalizers. Note that the scale parameter $\boldsymbol{\gamma}$ and shift parameter $\boldsymbol{\beta}$ in Eqn. (3) are omitted to simplify the diagram.

use softmax function to satisfy the summation constraint, $\sum_k \lambda_n^k = 1$. Compared with Eqn. (2) and Eqn. (3), the differences between SN and EN are three-fold. (1) The important ratios of mean and standard deviation in SN can be different, but such scheme is avoided in EN to ensure stability of training, because the learning capacity of EN already outperforms SN by learning different normalizers for different samples. (2) We use important ratios to combine the normalized feature maps instead of combining statistics of normalizers, reducing the bias in SN when combining the standard deviations. (3) Multiple $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ are adopted to re-scale and re-shift the normalized feature maps in EN.

To calculate the important ratios λ_n^k depended on the feature map of individual sample, we define

$$\boldsymbol{\lambda}_n = \mathcal{F}(\boldsymbol{X}_n, \boldsymbol{\Omega}; \Theta) \tag{4}$$

where $\boldsymbol{\lambda}_n = [\lambda_n^1, ..., \lambda_n^k, ...\lambda_n^K]$, and K is the total number of normalizers in EN. $\boldsymbol{\Omega}$ indicates a collection of statistics of different normalizers. We have $\boldsymbol{\Omega} = \{(\boldsymbol{\mu}^k, \boldsymbol{\delta}^k)\}_{k=1}^K$. $\mathcal{F}(\cdot)$ is a function (a small neural network) to calculate the instance-based important ratios, according to the input feature maps $\boldsymbol{X}_n$ and statistics $\boldsymbol{\Omega}$. Θ denotes learnable parameters of function $\mathcal{F}(\cdot)$. We carefully design a lightweight module to implement the function $\mathcal{F}(\cdot)$ in next subsection.

3.3. An Exemplar Normalization Layer

Fig. 2 shows a diagram of the key operations in an EN layer, including important ratio calculation and feature map normalization. Given an input tensor $\boldsymbol{X}$, a set of statistics $\boldsymbol{\Omega}$ are estimated. We use $\boldsymbol{\Omega}_k$ to denote the k-th statistics (mean and standard deviation). Then the EN layer uses $\boldsymbol{X}$ and $\boldsymbol{\Omega}$ to calculate the important ratios as shown in the right branch of Fig. 2 in blue. As shown in the left branch of

Fig. 2, multiple normalized tensors are also calculated.

In Fig. 2, there are three steps to calculate the important ratios for each sample. (1) The input tensor $\boldsymbol{X}$ is firstly down-sampled in the spatial dimension by using average pooling. The output feature matrix is denoted as $\boldsymbol{x} \in \mathbb{R}^{N\times C}$. Then we use every $\boldsymbol{\Omega}_k$ to pre-normalize $\boldsymbol{x}$ by subtracting the means and dividing by the standard deviations. There are K statistics and thus we have $\hat{\boldsymbol{x}} \in \mathbb{R}^{N\times K\times C}$. After that, a 1-D convolutional operator is employed to reduce the channel dimension of $\hat{\boldsymbol{x}}$ from C to C/r, which is shown in the first blue block in Fig. 2. Here r is a hyper-parameter that indicates the reduction rate. To further reduce the parameters in the above operation, we use group convolution with the group number C/r to ensure the total number of convolutional parameters always equals to C, irrelevant to the value of r. The output in this step is denoted as $\boldsymbol{z}$.

(2) The second step is to compute the pairwise correlation of different normalizers for each sample, which is motivated by the high-order feature representation [7, 4]. For the n-th sample, we use $\boldsymbol{z}_n \in \mathbb{R}^{K\times C}$ and its transposition $\boldsymbol{z}_n^T$ to compute the pairwise correlations by $\boldsymbol{v}_n = \boldsymbol{z}_n\boldsymbol{z}_n^T \in \mathbb{R}^{K\times K}$. Then $\boldsymbol{v}_n$ is reshaped to a vector to calculate the important ratios. Intuitively, the pairwise correlations capture the relationship between different normalizers for each sample, and allow the model to integrate more information to calculate the important ratios. In practice, we also find such operation could effectively stabilize the model training and make the model achieve higher performance.

(3) In the last step, the above vector $\boldsymbol{v}_n$ is firstly fed into a fully-connected (FC) layer followed by a tanh unit. This is to raise its dimensions to πK, where π is a hyper-parameter and the value of K is usually small, *e.g.* 3. In practice, we set the value of π as 50 in experiments. After that, we perform another FC layer to reduce the dimension to K. The output vector $\boldsymbol{\lambda}_n \in \mathbb{R}^{K\times 1}$ is regarded as the important ratios of the n-th sample for K normalizers, where each element is corresponding to an individual normalizer. Once we obtain the important ratio $[\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_N]^T$, the `softmax` function is applied to satisfy the summation constraint that the important ratios of different normalizers sum to 1.

Complexity Analysis. The numbers of parameters and computational complexity of different normalization methods are compared in Table 1. The additional parameters in EN are mainly from the convolutional and FC layers to calculate the data-dependant important ratios. In SN [23], such number is $2K$ since it adopts the global important ratios for both mean and standard deviant. In EN, the total number of parameters that is applied to generate the data-dependant important ratios is $C + \Psi(K)$, where C equals to the input channel size of the convolutional layer (*i.e.* "Conv." with C parameters in Fig. 2). $\Psi(K)$ is a function of K, which indicates the amount of parameters in the two FC layers (*i.e.* the top blue block in Fig. 2). In practice, since the number of K is small (*e.g.* $3 \sim 4$), the value of $\Psi(K)$ is just about $0.001M$. In this paper, EN employ a pool of normalizers that is the same as SN, *i.e.* {IN,LN,BN}. Thus the computational complexities of both SN and EN for estimating the statistics are $\mathcal{O}(NCHW)$. We also compare FLOPs in Sec. 4, showing that the extra #parameters of EN is marginal compared to SN, but its relative improvement over the ordinary BN is 300% larger than SN.

Table 1. **Comparisons** of parameters and computational complexity of different normalizers. $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ indicate the scale and shift parameters in Eqn.(2), and $\boldsymbol{\Theta}$ is the parameters of "Conv." and FC layer in proposed EN. K denotes the number of normalizer and $\Psi(\cdot)$ is a function of K that determines the number of $\boldsymbol{\Theta}$. $\{\omega_k, \nu_k\}_{k=1}^K$ are the learnable important ratios in SN [23].

Method	params	#params	computation complexity
BN [14]	$\boldsymbol{\gamma}, \boldsymbol{\beta}$	$2C$	$\mathcal{O}(NCHW)$
IN [36]	$\boldsymbol{\gamma}, \boldsymbol{\beta}$	$2C$	$\mathcal{O}(NCHW)$
LN [1]	$\boldsymbol{\gamma}, \boldsymbol{\beta}$	$2C$	$\mathcal{O}(NCHW)$
GN [39]	$\boldsymbol{\gamma}, \boldsymbol{\beta}$	$2C$	$\mathcal{O}(NCHW)$
BKN [37]	$\boldsymbol{A}$	C^2	$\mathcal{O}(NC^2HW)$
SN [23]	$\boldsymbol{\gamma}, \boldsymbol{\beta}, \{\omega_k, \nu_k\}_{k=1}^K$	$2C + 2K$	$\mathcal{O}(NCHW)$
EN	$\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{\Theta}$	$2KC+$ $C + \Psi(K)$	$\mathcal{O}(NCHW)$

4. Experiment

4.1. Image Classification with ImageNet dataset

Experiment Setting. We first examine the performance of proposed EN on ImageNet [6], a standard large-scale dataset for high-resolution image classification. Following [23], the $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ in all of the normalization methods are initialized as 1 and 0 respectively. In the training phase, the batch size is set as 128 and the data augmentation scheme is employed same as [9] for all of the methods. In inference, the single-crop validation accuracies based on 224×224 center crop are reported.

We use ShuffleNet v2 x0.5 [26] and ResNet50 [9] as the backbone network to evaluate various normalization methods since the difference in their network architectures and the number of parameters. Same as [26], ShuffleNet v2 is trained by using Adam optimizer with the initial learning rate 0.1. For ResNet50, all of the methods are optimized by using stochastic gradient decent (SGD) with stepwise learning rate decay. The hyper-parameter r in ShuffleNet v2 x0.5 and ResNet50 are set as 8 and 32 respectively since the smallest number of channels are different. The hyper-parameter π is 50. For fair comparison, we replace compared normalizers with EN in all of the normalization layers in the backbone network.

Result Comparison. Table 2 reports the efficiency and accuracy of EN against its counterparts including BN [14],

Table 2. Comparisons of classification accuracies (%), network parameters (Params.) and floating point operations per second (GFLOPs) of various methods on the validation set of ImageNet by using different network architectures.

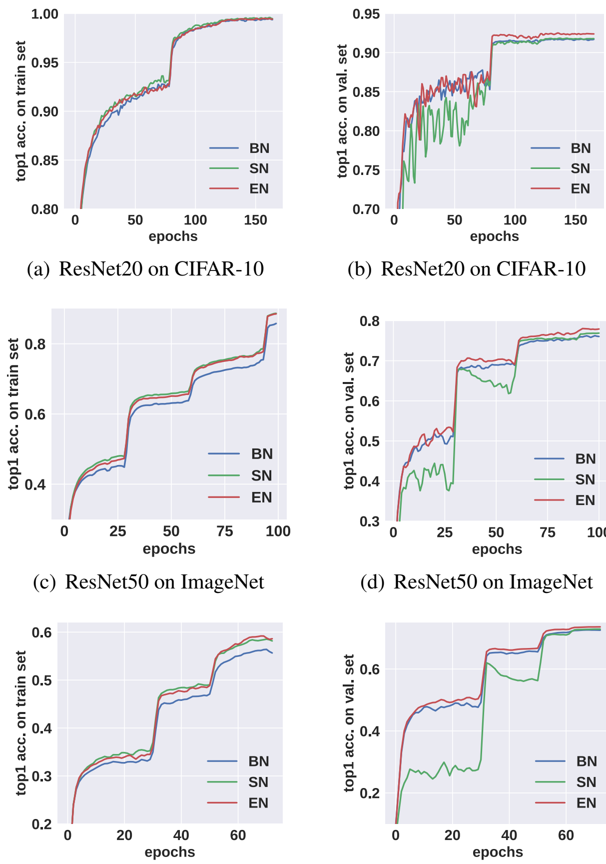
Backbone	Method	GFLOPs	Params.	top-1	top-5
ShuffleNet v2 x0.5	BN	0.046	1.37M	60.3	81.9
	SN	0.057	1.37M	61.2	82.9
	SSN	0.052	1.37M	61.2	82.7
	EN	0.063	1.59M	**62.2**	**83.3**
ResNet50	SENet	4.151	26.77M	77.6	93.7
	AANet	4.167	25.80M	77.7	**93.8**
	BN	4.136	25.56M	76.4	93.0
	GN	4.155	25.56M	76.0	92.8
	SN	4.225	25.56M	76.9	93.2
	SSN	4.186	25.56M	77.2	93.1
	EN	4.325	25.91M	**78.1**	93.6

GN [39], SN [23] and SSN [33]. For both two backbone networks, EN offers a super-performance and a competitive computational cost compared with previous methods. For example, by considering the sample-based ratio selection, EN outperforms SN 1.0%, and 1.2% on top-1 accuracy by using ShuffleNet v2 x0.5 and ResNet50 with only a small amount of GFLOPs increment. The top-1 accuracy curves of ResNet50 by using BN, SN and EN on training and validation set of ImageNet are presented in Fig. 3. We also compare the performance with state-of-the-art attention-based methods, *i.e.* SENet [11] and AANet [2], without bells and whistles, the proposed EN still outperforms these methods.

4.2. Noisy Classification with Webvision dataset

Experiment Setting. We also evaluate the performance of EN on noisy image classification task with Webvision dataset [18]. We adopt Inception v2 [35] and ResNet50 [9] as the backbone network. Since the smallest number of channels in Inception v2 is 32, the feature reduction rate r in the first "Conv." is set as 16 for such network architecture. In ResNet50 [9], we maintain the same reduction parameter $r = 32$ as Imagenet. The center crop with the image size 224×224 are adopted in inference. All of the models are optimized with SGD, where the learning rate is initialized as 0.1 and decreases at the iterations of $\{30, 50, 60, 65, 70\} \times 10^4$ with a factor of 10. The batch size is set as 256 and the data augmentation and data balance technologies are used by following [8]. In the training phase, we replace compared normalizers with EN in all of the normalization layers.

Result Comparison. Table 3 reports the top-1 and top-5 classification accuracies of various normalization methods. EN outperforms its counterparts by using both of two network architectures. Specially, by using ResNet50 as the backbone, EN significantly boost the top-1 accuracy from 72.8% to 73.5% compared with SN. It achieves about 3 times relative improvement of EN against SN compared to the ordinary plain ResNet50. Such performance gain is consistent with the results on ImageNet. The training and validation curves are shown in Fig. 3.



(a) ResNet20 on CIFAR-10 (b) ResNet20 on CIFAR-10
(c) ResNet50 on ImageNet (d) ResNet50 on ImageNet
(e) ResNet50 on Webvision (f) ResNet50 on Webvision

Figure 3. Top-1 training and validation accuracy curves of different normalization methods on CIFAR-10, ImageNet and Webvision dataset. Zoom in three times for the best view.

The cross dataset test is also conducted to investigate the transfer ability of EN since the categories in ImageNet and Webvision are the same. The model trained on one dataset is used to do the test on another dataset's validation set. The results are reported in Fig. 4 that EN still outperforms its counterparts.

4.3. Tiny Image Classification with CIFAR dataset

Experiment Setting. We also conduct the experiment on CIFAR-10 and CIFAR-100 dataset. The training batch size is 128. All of the models are trained by using the single GPU. The training process contains 165 epoches. The initial learning rate is set as 0.1 and decayed at 80 and 120 epoch, respectively. We also adopt the warm up scheme [9, 10] for

Table 3. Comparison of classification accuracies (%), network parameters and GFLOPs of various normalization methods on the validation set of Webvision by using different network architectures. The best results are bold.

Model	Norm	GFLOPs	Params.	top-1	top-5
Inception v2	BN	2.056	11.29M	70.7	88.0
	SN	2.081	11.30M	71.3	88.5
	EN	2.122	12.36M	**71.6**	**88.6**
ResNet50	BN	4.136	25.56M	72.5	89.1
	SN	4.225	25.56M	72.8	89.2
	EN	4.325	25.91M	**73.5**	**89.4**

Table 4. Top-1 and top-5 accuracy (%) of cross dataset results. The dataset before '$\rightarrow$' is adopted to train ResNet50 with various normalization methods. The validation set after '$\rightarrow$' is used for testing. The number of categories in two datasets are the same.

training set $\rightarrow$ val. set	method	top-1	top-5
ImageNet$\rightarrow$ Webvision	BN	67.9	85.8
	SN	68.0	86.3
	EN	**68.4**	**86.8**
Webvision $\rightarrow$ ImageNet	BN	64.4	84.3
	SN	61.1	81.0
	EN	**64.7**	**84.6**

Table 5. Top-1 accuracy (%) on CIFAR-10 and CIFAR-100 dataset by using various networks. The best results are bold.

Dataset	Backbone	BN	SN	EN
CIFAR-10	ResNet20	91.54	91.81	**92.41**
	ResNet56	93.15	93.41	**93.73**
	ResNet110	93.88	94.01	**94.22**
CIFAR-100	ResNet20	67.87	67.74	**68.78**
	ResNet56	70.83	70.70	**72.01**
	ResNet110	72.41	72.53	**73.32**

all of the models training, which increases the learning rate from 0 to 0.1 in the first epoch.

Result Comparison. The experiment results on CIFAR dataset are presented in Table 5. Compared with the previous methods, EN shows better performance than the other normalization methods over various depths of ResNet [9]. In particular, the top-1 accuracies of EN on CIFAR-100 are significantly improved by 1.04%, 1.31% and 0.79% compared with SN with different network depths.

4.4. Semantic Image Segmentation

Experiment Setting. We also evaluate the performance of EN on semantic segmentation task by using standard benchmarks, *i.e*. ADE20K [42] and Cityscapes [5] datasets, to demonstrate its generalization ability. Same as [23, 40], we use DeepLab [3] with ResNet50 as the backbone network and adopt the atrous convolution with the rate 2 and 4 in the last two blocks. The downsample rate of the backbone network is 8 and the bilinear operation is employed to upsample the predicted semantic maps to the size of the input image. All of the models are trained with 2 samples per G-PU by using "ploy" learning rate decay. The initial learning rate on ADE20K and Cityscapes are set as 0.02 and 0.01, respectively. Single-scale and multi-scale testing are used for evaluation. Note that the synchronization scheme is not used in SN and EN to estimate the batch mean and batch standard deviate across multi-GPU. To finetune the model on semantic segmentation, we use 8 GPU with 32 images per GPU to pre-train the EN-ResNet50 in ImageNet, thus we report the same configuration of SN (*i.e*. SN(8,32) [24]) for fair comparision.

Result Comparison. The mIoU scores on ADE20K validation set and Cityscapes test set are reported in Table 6. The performance improvement of EN is consistent with the results in classification. For example, the mIoUs on ADE20K and Cityscapes are improved from 38.4% and 75.8% to 38.9% and 76.1% by using multi-scale test.

Table 6. Semantic Segmentation results on ADE20K and C-ityscapes datasets. The backbone is ResNet50 with dilated convolutions. The subscripts "ss" and "ms" indicate single-scale and multi-scale test respectively. The best results are bold.

Method	ADE20K		Cityscapes	
	mIoU$_{ss}$	mIoU$_{ms}$	mIoU$_{ss}$	mIoU$_{ms}$
SyncBN	36.4	37.7	69.7	73.0
GN	35.7	36.6	68.4	73.1
SN	37.7	38.4	72.2	75.8
EN	**38.2**	**38.9**	**72.6**	**76.1**

Table 7. Top-1 accuracy (%) on ImageNet by using EN-ResNet50 with different ascending dimension hyper-parameter π.

Method	SN	EN (value of hyper-parameter π)				
		1	10	20	50	100
top-1	76.9	77.1	77.5	77.8	**78.1**	78.0
Δ *vs*. SN	-	+ 0.2	+ 0.6	+ 0.9	**+1.2**	+ 1.1

4.5. Ablation Study

Hyper-parameter π. We first investigate the effect of hyper-parameter π in Sec. 3.3. The top-1 accuracy on ImageNet by using ResNet50 as the backbone network are reported in Table 7. All of the EN models outperform SN. With the number of π increasing, the performance of classification growths steadily. The the gap between the lowest and highest is about 0.6% excluding $\pi = 1$, which demonstrates the model is not sensitive to the hyper-parameter π in most situations. To leverage the classification accuracy and computational efficiency, we set π as 50 in our model.

Hyper-parameter r. We also evaluate the differen-t group division strategy in the first "Conv." of Fig. 3.3 through controlling the hyper-parameter r. Although the

Table 8. Top-1 accuracy (%) on ImageNet by using EN-ResNet50 with different hyper-parameter r in the 'Conv.' of Sec. 3.3. Note that the total number of parameters with different r are the same.

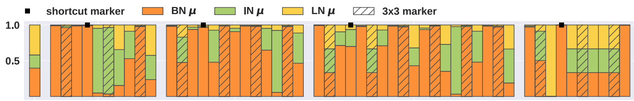
Method	SN	EN (value of hyper-parameter r)				
		2	4	16	32	64
top-1	76.9	77.7	77.9	77.9	**78.1**	77.7
Δ *vs.* SN	-	+0.8	+1.0	+1.0	**+1.2**	+0.8

Table 9. Top-1 and Top-5 accuracy (%) on ImageNet by using EN-ResNet50 with different configurations.

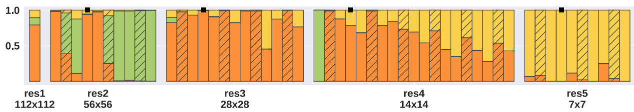
Method	top-1 / top5	top-1 / top5 Δ *vs.* EN
EN-ResNet50	78.1 / 93.6	-
$a.\rightarrow$ 2-layer MLP	76.7 / 92.9	-1.4 / -0.7
$b.\rightarrow$ w/o Conv.	77.6 / 92.9	-0.5 / -0.7
$c.\rightarrow$ ReLU	77.7 / 93.4	-0.4 / -0.2
$d.\rightarrow$ single γ, β	77.6 / 93.3	-0.5 / -0.3

total numbers of parameters in "Conv." layer are the same by using distinct r, the reduced feature dimensions are different, leading to the different computational complexity, *i.e.* the larger r, the smaller computation cost in the subsequent block. Table 8 shows the top-1 accuracy on ImageNet by using EN-ResNet50 with different group division in the first "Conv." shown in Fig. 2. All of the configurations achieve higher performance than SN. With the value of r growths, the performance of EN-ResNet50 increases stably expect 64, which equals to the smallest number of channels in ResNet50. These results indicate that feature dimension reduction benefits to the performance increment. However, such advantage may disappear if the reduction rate equals to the smallest number of channels.

Other Configurations. We replace the other components in the EN layer to verify their effectiveness. The configurations for comparison are as follows. (a) A 2-layer multi-layer perceptron (MLP) is used to replace the designed important ratio calculation module in Fig. 2. The MLP reduces the feature dimension to $1/32$ in the first layer followed by an activation function, and then reduce the dimension to the number of important ratios in the second layer. (b) The "Conv." operation in the Fig. 2 are omitted and pairwise correlations $\boldsymbol{v}_n$ in Sec. 3.3 'step(2)' are directly computed. (c) The Tanh activation function in the top blue block of Fig. 2 is replaced with ReLU. (d) Instead of multiple γ, β in Eqn. (3) (*i.e.* each γ, β is corresponding to one normalizer), single γ, β are adopted. Table 9 reports the comparisons of proposed EN with different internal configuration. According to the results, the current configuration of EN achieves the best performance compared with the other variants. It is worthy to note that we find the output of 2-layer MLP changing dramatically in the training phase (*i.e.* important ratios), making the distribution of



(a) The average ratios on ImageNet validation set



(b) The average ratios on Webvision validation set

Figure 4. The averaged sample ratios in different layers of ResNet50 on ImageNet and Webvision validation set. The y-axis denotes the important ratios of different normalizers after the softmax operation (*i.e.* sum to 1). The x-axis shows different residual blocks of ResNet50 and the image resolution in each block is represented as well. Different datasets learn distinct averaged ratios for different normalizers in different layers of the network.

feature maps at different iterations changed too much and leading to much poor accuracy.

4.6. Analysis of EN

Learning Dynamic of Ratios on Dataset. Since the parameters which are adopted to learn the important ratios $\boldsymbol{\lambda}$ in EN layer are initialized as 0, the important ratios of each sample in each layer have uniform values (*i.e.* $1/3$) at the beginning of the model training. In the training phase, the values of $\boldsymbol{\lambda}$ changes between 0 and 1. We first investigate the averaged sample ratios in different layers of ResNet50 on ImageNet and Webvision validation set. We use the optimized model to calculate the ratios of each sample in each layer, then the average ratios of each layer are calculated over all of the validation set. According to Fig. 4, once the training dataset is determined, the learned averaged ratios are usually distinct for different datasets.

To analysis the changes of ratios in the training process, Fig. 5 plots the leaning dynamic of ratios of 100 epochs for 53 normalization layers in ResNet50 . Each value of ratios are averaged over all of the samples in ImageNet validation set. From the perspective of the entire dataset, the changes of ratios in each layer of EN are similar to those in SN, whose values have smooth fluctuation in the training phase, implying that distinct layers may need their own preference of normalizers to optimize the model in different epochs.

Learning Dynamic of Ratios on Classes and Images. One advantage of EN compared with SN is able to learn important ratios to adaptive to different exemplars. To illustrate such benefit of EN, we further plot the averaged important ratios of different classes (*i.e.* w/ and w/o similar appearance) in different layers in Fig. 6, as well as the important ratios of various image samples in different layers in Fig. 7. We have the following observations.

(1) Different classes learn their own important ratios in different layers. However, once the neural network is opti-

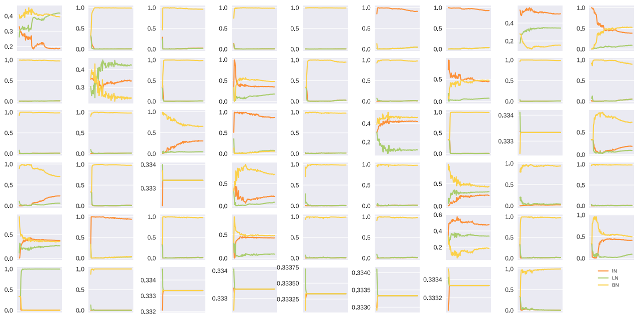


Figure 5. The visualization of averaged sample ratios in 53 normalization layers of EN-ResNet50 trained on ImageNet for 100 epoches. The y-axis of each sub-figure denotes the important ratios of different normalizers. The x-axis shows the different training epoches. Zoom in three times for the best view.

mized on a certain dataset (*e.g*. ImageNet), the trend of the ratio changes of are similar in different epochs. For example, in Fig. 6, since the Persian cat and Siamese cat have a similar appearance, their leaned ratio curves are very close and even coincident in some layers, *e.g*. Layer5 and Layer 10. While the ratio curves from the class of Cheeseburger are far away from the above two categories. But in most layers, the ratio changes of different normalizers are basically the same, only have the numerical nuances.

(2) For the images with the same class index but various appearances, their learned ratios could also be distinct in different layers. Such cases are shown in Fig. 7. All of the images are sampled from confectionery class but with various appearance, *e.g*. the exemplar of confectionery and shelves for selling candy. According to Fig. 7, different images from the same category also obtained different ratios in bottom, middle and top normalization layers.

5. Conclusion

In this paper, we propose Exemplar Normalization to learn the linear combination of different normalizers with a sample-based manner in a single layer. We show the effectiveness of EN on various computer vision tasks, such as classification, detection and segmentation, demonstrate its superior learning and generalization ability than static learning-to-normalize method such as SN. In addition, the interpretable visualization of learned important ratios reveals the properties of classes and datasets. The future work will explore EN in more intelligent tasks. In addition, the task-oriented constraint on the important ratios will also be a potential research direction.

Acknowledgement This work was partially supported by No. 2018YFB1800800, Open Research Fund from Shenzhen Research Institute of Big Data No. 2019ORF01005, 2018B030338001, 2017ZT07X152, ZDSYS201707251409055, HKU Seed Fund for Basic Research and Start-up Fund.

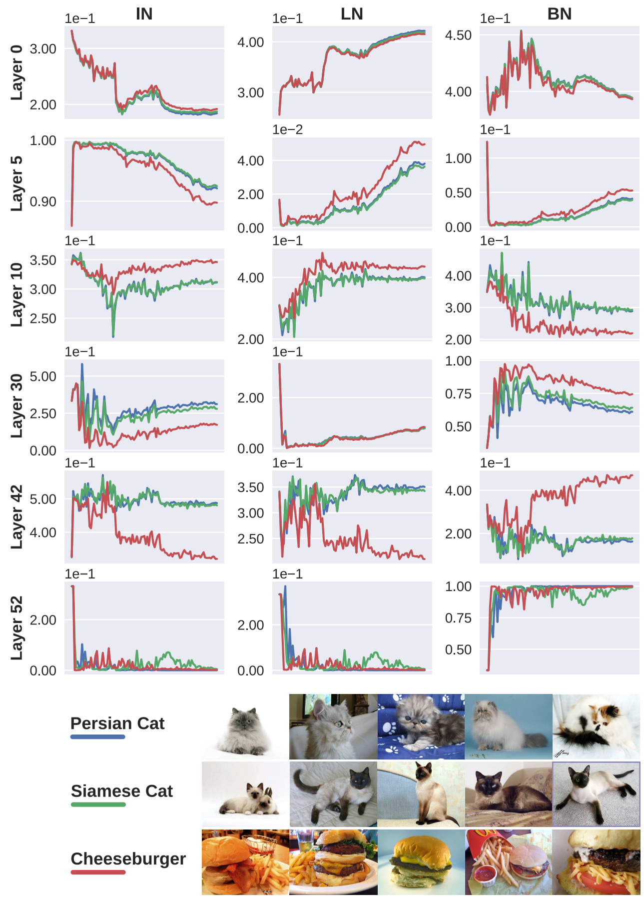


Figure 6. The visualization of the important ratios of 3 categories (*i.e*. Persian cat, Siamese cat and Cheeseburger) in 6 different layers of ResNet50. Each column indicates one of the normalizers.

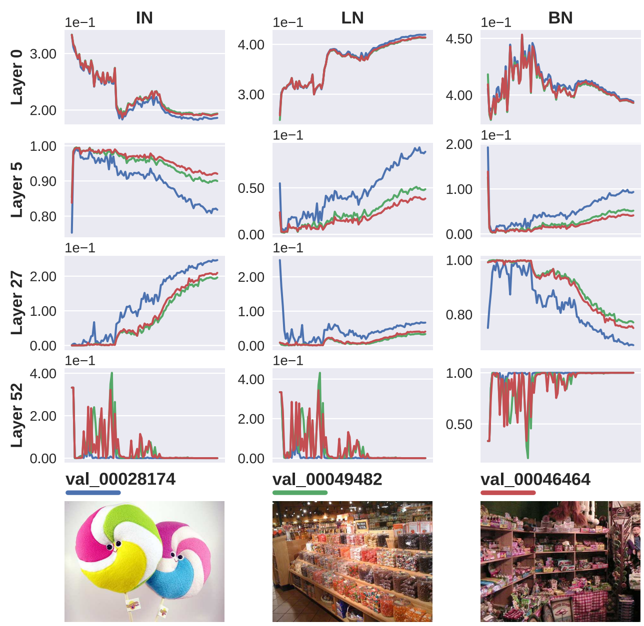


Figure 7. The visualization of the important ratios of 3 samples selected from Confectionery class in different layers of ResNet50.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

[2] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *ICCV*, 2019.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[4] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ 2-nets: Double attention networks. In *NeurIPS*, 2018.

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[7] Zilin Gao, Jiangtao Xie, Qilong Wang, and Peihua Li. Global second-order pooling convolutional networks. In *CVPR*, 2019.

[8] Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R Scott, and Dinglong Huang. Curriculumnet: Weakly supervised learning from large-scale web images. In *ECCV*, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[12] Lei Huang, Xianglong Liu, Yang Liu, Bo Lang, and Dacheng Tao. Centered weight normalization in accelerating training of deep neural networks. In *ICCV*, 2017.

[13] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *NeurIPS*, 2017.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[15] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NeurIPS*, 2016.

[16] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial intelligence and statistics*, 2016.

[17] Boyi Li, Felix Wu, Kilian Q. Weinberger, and Serge J. Belongie. Positional normalization. *arXiv:1907.04312*, 2019.

[18] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv:1708.02862*, 2017.

[19] Xilai Li, Wei Sun, and Tianfu Wu. Attentive normalization. *arXiv:/1908.01259*, 2019.

[20] Qianli Liao, Kenji Kawaguchi, and Tomaso Poggio. Streaming normalization: Towards simpler and more biologically-plausible normalizations for online and recurrent learning. *arXiv:1610.06160*, 2016.

[21] Iain Murray Lucas Deecke and Hakan Bilen. Mode normalization. In *ICLR*, 2019.

[22] Ping Luo, Zhanglin Peng, Jiamin Ren, and Ruimao Zhang. Do normalization layers in a deep convnet really need to be distinct? *arXiv:1811.07727*, 2018.

[23] Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization. In *ICLR*, 2019.

[24] Ping Luo, Ruimao Zhang, Jiamin Ren, Zhanglin Peng, and Jingyu Li. Switchable normalization for learning-to-normalize deep representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[25] Ping Luo, Peng Zhanglin, Shao Wenqi, Zhang Ruimao, Ren Jiamin, and Wu Lingyun. Differentiable dynamic normalization for learning deep representation. In *ICML*, 2019.

[26] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: practical guidelines for efficient CNN architecture design. In *ECCV*, 2018.

[27] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[28] Hyeonseob Nam and Hyo-Eun Kim. Batch-instance normalization for adaptively style-invariant neural networks. In *NeurIPS*, 2018.

[29] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: enhancing learning and generalization capacities via ibn-net. In *ECCV*, 2018.

[30] Xingang Pan, Xiaohang Zhan, Jianping Shi, Xiaoou Tang, and Ping Luo. Switchable whitening for deep representation learning. In *ICCV*, 2019.

[31] Ethan Perez, Harm de Vries, Florian Strub, Vincent Dumoulin, and Aaron C. Courville. Learning visual reasoning without strong priors. *arXiv:1707.03017*, 2017.

[32] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.

[33] Wenqi Shao, Tianjian Meng, Jingyu Li, Ruimao Zhang, Yudian Li, Xiaogang Wang, and Ping Luo. Ssn: Learning sparse switchable normalization via sparsestmax. In *CVPR*, 2019.

[34] Wenqi Shao, Shitao Tang, Xingang Pan, Ping Tan, Xiaogang Wang, and Ping Luo. Channel equilibrium networks for learning deep representation. *arXiv:2003.00214*, 2020.

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

[36] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv:1607.08022*, 2016.

[37] Guangrun Wang, Jiefeng Peng, Ping Luo, Xinjiang Wang, and Liang Lin. Batch kalman normalization: Towards training deep neural networks with micro-batches. 2018.

[38] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *CVPR*, 2020.

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

[40] Ruimao Zhang, Wei Yang, Zhanglin Peng, Pengxu Wei, Xiaogang Wang, and Liang Lin. Progressively diffused networks for semantic visual parsing. *Pattern Recognition*, 90:78–86, 2019.

[41] Zhaoyang Zhang, Jingyu Li, Wenqi Shao, Zhanglin Peng, Ruimao Zhang, Xiaogang Wang, and Ping Luo. Differentiable learning-to-group channels via groupable convolutional neural networks. In *ICCV*, 2019.

[42] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017.