Between-class Learning for Image Classification

Yuji Tokozume[1] Yoshitaka Ushiku[1] Tatsuya Harada[1,2]
[1]The University of Tokyo [2]RIKEN
{tokozume,ushiku,harada}@mi.t.u-tokyo.ac.jp

Abstract

In this paper, we propose a novel learning method for image classification called Between-Class learning (BC learning)[1]*. We generate between-class images by mixing two images belonging to different classes with a random ratio. We then input the mixed image to the model and train the model to output the mixing ratio. BC learning has the ability to impose a constraint on the shape of the feature distributions, and thus the generalization ability is improved. BC learning is originally a method developed for sounds, which can be digitally mixed. Mixing two image data does not appear to make sense; however, we argue that because convolutional neural networks have an aspect of treating input data as waveforms, what works on sounds must also work on images. First, we propose a simple mixing method using internal divisions, which surprisingly proves to significantly improve performance. Second, we propose a mixing method that treats the images as waveforms, which leads to a further improvement in performance. As a result, we achieved* 19.4% *and* 2.26% *top-1 errors on ImageNet-1K and CIFAR-10, respectively.*

1. Introduction

Deep convolutional neural networks (CNNs) [18] have achieved high performance in various tasks, such as image recognition [17, 11], speech recognition [1, 21], and sound recognition [19, 26]. One of the biggest themes of research on image recognition has been network engineering. Many types of networks have been proposed mainly in ILSVRC competition [17, 23, 25, 11, 28, 30, 12]. Furthermore, training deep neural networks is difficult, and many techniques have been proposed to achieve a high performance: data augmentation techniques [17], new network layers such as dropout [24] and batch normalization [14], optimizers such as Adam [15], and so on. Thanks to these research studies, training deep neural networks has become relatively easy, with a stable performance, at least for image classification. At present, a novel approach is needed for further improvement.

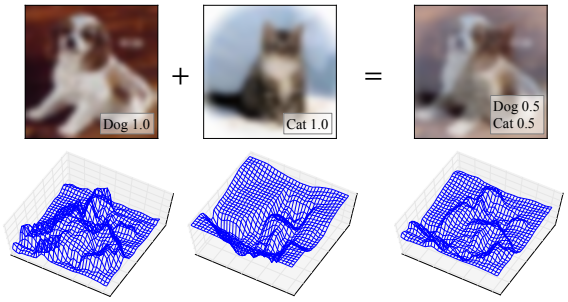


Figure 1. We argue that CNNs have an aspect of treating the input data as waveforms. In that case, a mixture of two images is a mixture of two waveforms. This make sense *for machines*, although it does not visually make sense *for humans*.

In [3] (anonymized parallel submission), we are proposing a simple and powerful novel learning method named *Between-Class learning (BC learning)* for deep sound recognition. BC learning aims to learn a classification problem by solving the problem of predicting the mixing ratio between two different classes. We generated between-class examples by mixing two sounds belonging to different classes with a random ratio. We then input the mixed sound to the model and trained the model to output the mixing ratio. The advantages of BC learning are not limited only to the increase in variation of the training data. We also argue that BC learning has the ability to impose a constraint on the feature distributions, which cannot be achieved with standard learning, and thus the generalization ability is improved. We carefully designed the method of mixing two sounds, considering the difference in the sound energies, to achieve a satisfactory performance. As a result, BC learning improves the performance on various sound recognition networks, datasets, and data augmentation schemes, and we achieved a performance surpasses the human level in sound classification tasks.

The question here is whether BC learning also performs well on images. The core idea of BC learning itself, i.e., mixing two examples and training the model to output the

[1]Similar idea *mixup* [29] was proposed on Oct. 25, 2017 (unpublished), but our preliminary experimental results on CIFAR-10 and ImageNet-1K were already presented in ILSVRC2017 on July 26, 2017.

mixing ratio, can be used irrespective of the modality of input data. BC learning is applicable to sounds, because sound is a kind of wave motion and a mixture of multiple sound data still counts as a sound. However, an image is not a kind of wave motion, and mixing multiple image data does not visually make sense. We show an example of a mixed image in Fig. 1(top). A mixed image loses its objectness and does not count as an image. Therefore, it appears inappropriate to apply BC learning to images.

However, the important thing is not how *humans* perceive the mixed data, but how *machines* perceive them. We argue that CNNs have an aspect of treating input data as waveforms, considering the recent studies on speech and sound recognition and the characteristics of image data as pixel values. We assume that CNNs recognize images by treating them as waveforms in quite a similar manner to how they recognize sounds. Thus, a mixture of two images is a mixture of two waveforms as shown in Fig. 1(bottom), and it would make sense *for machines*. Therefore, what is effective for sounds would also be effective for images.

We thus propose BC learning for images in this paper. First, we propose the simplest mixing method using internal divisions. Surprisingly, this mixing method proves to perform well. Second, we also propose a new mixing method that treats images as waveform data. Before normalizing with the mean and standard deviation of the whole training data, we subtract the per-image mean value and make each image of zero-mean. We then define the image energy as the standard deviation per image after normalizing, and mix two images considering the image energy in quite a similar manner to what we did for sounds. This mixing method is also simple and easy to implement, and leads to a further improvement in performance.

Experimental results show that BC learning also improves the performance of various types of image recognition networks from a simple network to the state-of-the-art networks. The top-1 error of ResNeXt-101 (64×4d) [28] on ImageNet-1K is improved from 20.4% to 19.4% by using the simplest BC learning. Moreover, the error rate of the state-of-the-art Shake-Shake Regularization [9] on CIFAR-10 dataset is improved from 2.86% to 2.26% by using the improved BC learning. Finally, we visualize the learned features and show that BC learning indeed imposes a constraint on the feature distribution. The contributions of this paper are as follows:

- We applied BC learning to images by mixing two images belonging to different classes and training the model to output the mixing ratio.
- We argued that CNNs have an aspect of treating input data as waveforms, and proposed a mixing method that treats the images as waveforms.
- We conducted experiments extensively and demonstrated the effectiveness of BC learning.

The remainder of this paper is organized as follows. In Section 2, we provide a summary of BC learning for sound recognition as a related work, which we are proposing in [3]. We then propose BC learning for image recognition in Section 3, explaining the relationship with BC learning for sounds. In Section 4, we compare the performance of standard and BC learning, and demonstrate the effectiveness of BC learning. Finally, we conclude our paper in Section 5.

2. BC learning for sounds

In this section, we describe our BC learning for sound recognition as a related work. The contents in this section is a summarization of Section 3 of [3]. Please see [3] for more detailed information.

2.1. Overview

In standard learning for classification problems, we select a single training example from the dataset and input it to the model. We then train the model to output a one-hot class label. By contrast, in BC learning, we select two training examples from different classes and mix these two examples using a random ratio. We then input the mixed data to the model and train the model to output the mixing ratio. BC learning uses only mixed data and labels, and thus never uses pure data and labels.

2.2. Mixing method

The method of mixing two sounds greatly affects the performance. Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two sounds belonging to different classes randomly selected from the training dataset, and $\mathbf{t}_1$ and $\mathbf{t}_2$ be their one-hot labels. We generated a random ratio r from the uniform distribution $U(0,\ 1)$, and mixed two sets of data and labels with this ratio. We mixed two labels simply by $r\,\mathbf{t}_1 + (1-r)\,\mathbf{t}_2$, because we aimed to train the model to output the mixing ratio. We then explained how to mix $\mathbf{x}_1$ and $\mathbf{x}_2$, which should be carefully designed to achieve a satisfactory performance. The simplest method is $r\,\mathbf{x}_1 + (1-r)\,\mathbf{x}_2$. Here, $\frac{r\,\mathbf{x}_1+(1-r)\,\mathbf{x}_2}{\sqrt{r^2+(1-r)^2}}$ is better because sound energy is proportional to the square of the amplitude. However, auditory perception of a sound mixed with this method would not be $\mathbf{x}_1 : \mathbf{x}_2 = r : (1-r)$ if the difference of the sound pressure level of $\mathbf{x}_1$ and $\mathbf{x}_2$ is large. In this case, training the model with a label of $\{r,\ 1-r\}$ is inappropriate. We then considered using a new coefficient $p(r,\ G_1,\ G_2)$ instead of r, and mixed two sounds by $\frac{p\,\mathbf{x}_1+(1-p)\,\mathbf{x}_2}{\sqrt{p^2+(1-p)^2}}$, where G_1 and G_2 is the sound pressure level of $\mathbf{x}_1$ and $\mathbf{x}_2$ [dB], respectively. We defined p so that the auditory perception of the mixed sound becomes $r : (1-r)$. We hypothesized that the ratio of auditory perception for the network is the same as the ratio of amplitude and then solved $p \cdot 10^{\frac{G_1}{20}} : (1-p) \cdot 10^{\frac{G_2}{20}} = r : (1-r)$.

Finally, we proposed a mixing method as follows:

$$mix_r(\mathbf{x}_1,\, \mathbf{x}_2) = \frac{p\,\mathbf{x}_1 + (1-p)\,\mathbf{x}_2}{\sqrt{p^2 + (1-p)^2}}, \quad \text{where} \;\; p = \frac{1}{1 + 10^{\frac{G_1 - G_2}{20}} \cdot \frac{1-r}{r}}. \tag{1}$$

2.3. How BC learning works

We argued that BC learning has the ability to provide a constraint to the feature distribution, which cannot be achieved with the standard learning, and thus the generalization ability is improved. We hypothesized that a mixed sound $mix_r(\mathbf{x}_1, \mathbf{x}_2)$ is projected onto the point near the internally dividing point of the original two features. This hypothesis came from the fact that linearly-separable features are learned in a hidden layer close to the output layer of deep neural networks [2] and that humans can perceive which of the two sounds is louder or softer from the digitally mixed sound. It is expected that an internally dividing point of the input space almost corresponds to that of the high-level feature space, at least for sounds. We visualized the feature distributions of the standard-learned model using PCA and showed that this hypothesis was indeed correct. Then, we argued that BC learning has the following two effects under this hypothesis.

Enlargement of Fisher's criterion. We argued that BC learning leads to an enlargement of Fisher's criterion [8] (*i.e.*, the ratio of the between-class distance to the within-class variance) in the feature space. We explain the reason in Fig. 2(top). If Fisher's criterion is small, the feature distribution of the mixed sounds becomes large, and would have a large overlap with one or both feature distributions of class A and class B (Fig. 2(upper left)). In this case, some mixed sounds are projected onto one of the classes as shown in this figure, and the model cannot output the mixing ratio. BC learning gives a penalty to this situation because BC learning trains a model to output the mixing ratio. To let the model output the mixing ratio and make the penalty of BC learning small, Fisher's criterion should be large as shown in Fig. 2(upper right). In this case, the overlap becomes small and the model becomes able to output the mixing ratio, and BC learning gives a small penalty. Therefore, BC learning enlarges Fisher's criterion between any two classes in the feature space.

Regularization of positional relationship among feature distributions. We expected that BC learning also has the effect of regularizing the positional relationship among class feature distributions. In standard learning, there is no constraint on the positional relationship among classes as long as the features of two classes are linearly separable.

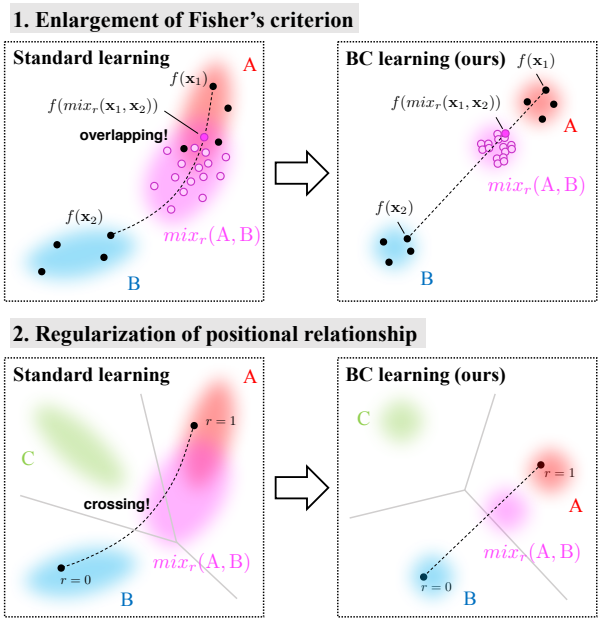


Figure 2. We argued that BC learning has the ability to provide a constraint to the feature distribution, which cannot be achieved with the standard learning. This figure represents the class distribution in the feature space. The black dashed line represents the trajectory of the feature when we input a mixture of two particular sounds to the model changing the mixing ratio from 0 to 1.

We found that a standard-learned model sometimes misclassifies a mixed sound of class A and class B as a class other than A or B. This is an undesirable situation because there is little possibility that a mixed sound of two classes becomes a sound of other classes. In this case, we assume that the features of each class are distributed as depicted in Fig. 2(lower left). The decision boundary of class C appears between class A and class B, and the trajectory of the features of the mixed sounds crosses the decision boundary of class C.

BC learning can avoid the situation in which the decision boundary of another class appears between two classes, because BC learning trains a model to output the mixing ratio instead of misclassifying the mixed sound as different classes. By doing this, we assumed that the features of each class become distributed as in shown Fig. 2(lower right). The feature distributions of the three classes form an acute-angled triangle, and the decision boundary of class C does not appear between class A and class B. Therefore, we argued that BC learning has the ability to provide a constraint to the feature distribution, and thus BC learning improves the generalization ability.

3. From sounds to images

In this section, we consider applying BC learning to images. Following BC learning for sounds [3], we select two

training examples from different classes and mix these two examples using a random ratio. We then input the mixed data to the model and train the model to output the mixing ratio. BC learning uses only mixed data and labels and, thus, never uses pure data and labels. How to mix two examples is also important for images. First, we propose the simplest mixing method in Section 3.1. Second, we discuss why BC learning can also be applied to images in Section 3.2. Finally, in Section 3.3, we propose a better version of BC learning, considering the discussion in 3.2.

3.1. Simple mixing

Let $\mathbf{x}_1$ and $\mathbf{x}_2$ be two images belonging to different classes randomly selected from the training dataset, and $\mathbf{t}_1$ and $\mathbf{t}_2$ be their one-hot labels. Note that $\mathbf{x}_1$ and $\mathbf{x}_2$ may have already been preprocessed or applied to data augmentation, and they have the same size as that of the input of the network. We generate a random ratio r from $U(0,\ 1)$, and mix two sets of data and labels with this ratio. We mix two labels simply by $r\,\mathbf{t}_1 + (1-r)\,\mathbf{t}_2$, because we aim to train the model to output the mixing ratio. We now explain how to mix $\mathbf{x}_1$ and $\mathbf{x}_2$. In [3], we proposed a carefully designed mixing method of two sounds considering the difference in the sound energies, as we mentioned in Section 2.2. In a sound data, 0 is the absolute center, and the distance from 0 represents the sound energy. However, the pixel values of an image data do not have an absolute center, and there appears to be no concept of energy. We thus first propose the following mixing method as the simplest method using internal divisions[2]:

$$mix_r(\mathbf{x}_1,\ \mathbf{x}_2) = r\,\mathbf{x}_1 + (1-r)\,\mathbf{x}_2. \tag{2}$$

Surprisingly, this mixing method performs well.

3.2. Why BC learning works on images

BC learning is applicable to sounds because a mixed sound still counts as a sound. Sound is a kind of wave motion, and mixing two sounds physically makes sense. Humans can recognize two sounds and perceive which of the two sounds is louder or softer from the digitally mixed sound. However, image data, as pixel values, is not a kind of wave motion *for humans*. Therefore, mixing multiple images does not visually make sense.

However, the important thing is not whether mixing two data physically makes sense, or whether humans can perceive a mixed data, but how a machine perceives a mixed data. We argue that CNNs have an aspect of treating input data as waveforms. In fact, recent studies have demonstrated that CNNs can learn speech and sounds directly from raw waveforms, and each filter learns to respond to a particular frequency area [21, 26, 7]. It is also known that images, as pixel values, can be transformed into components

[2] *mixup* [29] used this mixing method.

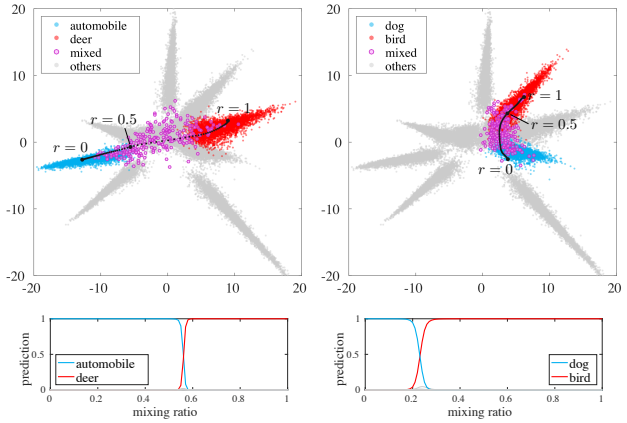


Figure 3. Visualization of the feature space of a standard-learned model using PCA. The features of the mixed images are distributed between two classes.

of various frequency areas by using 2-D Fourier transform [22], and some convolutional filters can act as frequency filters (*e.g.*, a Laplacian filter acts as a high-pass filter [5]). Therefore, it is expected that each convolutional filter of a CNN learns to extract the frequency features. In this way, we assume that CNNs recognize images by treating them as waveforms in quite a similar manner to how they recognize sounds. Thus, because a mixture of two images is a mixture of two waveforms *for machines*, what is effective for sounds would also be effective for images.

We visualized the feature distribution of the standard-learned model against mixed data using PCA. We used the output of the 10-th layer of an 11-layer CNN trained on CIFAR-10. We mixed two images using Eqn. 2 and the results are shown in Fig. 3. The magenta dots represent the feature distribution of the mixed images of *automobile*, *deer* (Left), *dog* and *bird* (Right) with a ratio of $0.5 : 0.5$, and the black dotted line represents the trajectory of the feature when we input a mixture of two particular images to the model, changing the mixing ratio from 0 to 1. This figure shows that the mixture of two images is projected onto the point near the internally dividing point of two features, and the features of the mixed images are distributed between two classes, which is the same tendency observed for sounds [3]. Therefore, the same effect as BC learning for sounds, i.e., an enlargement of Fisher's criterion in the feature space and a regularization of the positional relationship among the feature distributions of the classes, is expected. We compare the feature distributions learned with standard and BC learning and demonstrate that a different shape of feature distribution is learned with BC learning in the visualization experiment.

3.3. BC+: Images as waveform data

Here, we consider a new mixing method, which treats images as waveform data. We regard image data as a 2-D

waveform consisting of (R, G, B) vectors. In recent state-of-the-art methods, the input data is normalized for each channel using the mean and standard deviation calculated from the whole training data [28, 13, 9]. In this case, the mean of each image is not equal to 0, and each image data $\mathbf{x}_i$ is represented as $\mathbf{x}_i = \mu_i + \mathbf{d}_i$, where μ_i and $\mathbf{d}_i$ are the static component and wave component, respectively. Here, the simplest mixing method of Eqn. 2 can be rewritten as $mix_r(\mathbf{x}_1, \mathbf{x}_2) = \{r\,\mu_1 + (1-r)\,\mu_2\} + \{r\,\mathbf{d}_1 + (1-r)\,\mathbf{d}_2\}$. We assume that the performance improvement with Eqn. 2 is mainly owing to the wave component $r\,\mathbf{d}_1 + (1-r)\,\mathbf{d}_2$ if CNNs treat the input data as waveforms. Moreover, the static component $r\,\mu_1 + (1-r)\,\mu_2$ can have a bad effect because mixing two waveforms generally hypothesizes that the static components of two waveforms are same.

We then remove the static component by subtracting the per-image mean value (not channel-wise mean) before normalizing with the mean and standard deviation of the whole training data. Now we can treat each image as a zero-mean waveform and apply the same schemes as we did to sounds as described in Section 2.2. First, we consider mixing two images with:

$$mix_r(\mathbf{x}_1, \mathbf{x}_2) = \frac{r\,\mathbf{x}_1 + (1-r)\,\mathbf{x}_2}{\sqrt{r^2 + (1-r)^2}} \tag{3}$$

instead of $r\,\mathbf{x}_1 + (1-r)\,\mathbf{x}_2$, considering that waveform energy is proportional to the square of the amplitude. This process prevents the input variance from decreasing.

Second, we take the difference of energies into consideration, in order to make the perception of the mixed image $r : (1-r)$. We use a new coefficient $p(r, G_1, G_2)$ instead of r, and mix two images by $\frac{p\,\mathbf{x}_1 + (1-p)\,\mathbf{x}_2}{\sqrt{p^2 + (1-p)^2}}$, where G_1 and G_2 are the image energies. We define the image energy as the square of standard deviation per image (σ_1^2 and σ_2^2). We hypothesize that the ratio of image perception for the network is the same as the ratio of amplitude because the main component functions of CNNs, such as conv/fc, relu, max pooling, and average pooling, satisfy homogeneity (*i.e.*, $f(\alpha\,\mathbf{x}) = \alpha f(\mathbf{x})$) if we ignore the bias. We then solve $p\,\sigma_1 : (1-p)\,\sigma_2 = r : (1-r)$. Finally, we obtain the proposed mixing method:

$$\begin{aligned} mix_r(\mathbf{x}_1, \mathbf{x}_2) &= \frac{p\,\mathbf{x}_1 + (1-p)\,\mathbf{x}_2}{\sqrt{p^2 + (1-p)^2}}, \\ \text{where}\quad p &= \frac{1}{1 + \frac{\sigma_1}{\sigma_2} \cdot \frac{1-r}{r}}. \end{aligned} \tag{4}$$

The essential difference from the mixing method for sounds (Eqn.1) is only the definition of energies. This mixing method is also easy to implement, and experimentally proves to lead to a further improvement in performance, compared to the simplest mixing method of Eqn. 2.

Table 1. Results of ResNeXt-101 (64 × 4d) [28] on ImageNet-1K dataset. BC learning improves the performance when using the default learning schedule. Furthermore, the performance is further improved when using a longer learning schedule, and the single-crop top-1 error is improved by around 1% compared to the default performance reported in [28].

# epochs	Learning	Top-1/top-5 val. error (%) Single-crop	10-crop
100	Standard	20.4 / 5.3 [28]	18.90 / 4.61
	BC (ours)	19.92 / 4.91	18.66 / 4.26
150	Standard	20.44 / 5.25	18.98 / 4.43
	BC (ours)	**19.43 / 4.80**	**18.22 / 4.13**

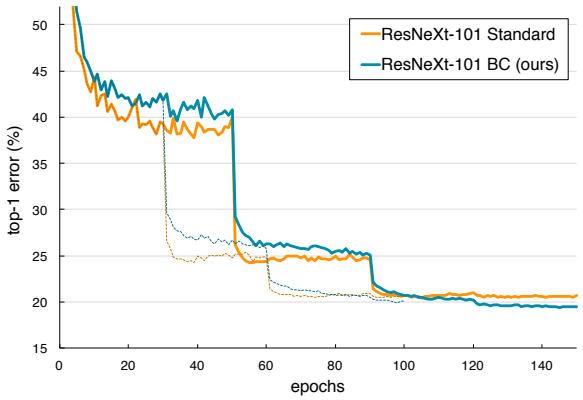


Figure 4. Training curves of ResNeXt-101 (64 × 4d) on ImageNet-1K dataset. The dashed lines represent the training curves when using the default learning schedule, and the solid lines represent the training curves when using a longer learning schedule.

4. Experiments

4.1. Experiments on ImageNet-1K

First, we compare the performance of standard and BC learning on the 1,000-class ImageNet classification task [20]. In this experiment, we used the simple BC learning. We selected ResNeXt-101 (64 × 4d) [28] as the model for training because it has a state-of-the-art level performance and, moreover, the official Torch [6] training codes are available[3]. To validate the comparison, we incorporated BC learning into these official codes. When using BC learning, we selected two training images and applied the default data augmentation scheme as in described [28] to each image, and obtained two 224 × 224 images. We then mixed these two images with a random ratio selected from $U(0, 1)$. In addition to the default learning schedule (# of epochs = 100), we also tried a longer learning schedule (# of epochs = 150). In 100-epochs training, we started training with a learning rate of 0.1 and then divided the learning rate by 10 at the epoch in {30, 60, 90}, as in [28]. In 150-

[3]https://github.com/facebookresearch/ResNeXt

Table 2. Results on CIFAR-10 and CIFAR-100 datasets. We show the average and the standard error of 5 or 10 trials. BC learning improves the performance of various settings. Note that † is trained with a different learning setting from the default.

Model	Learning	Error rate (%) on CIFAR-10	Error rate (%) on CIFAR-100
11-layer CNN	Standard	6.07 ± 0.04	26.68 ± 0.09
	BC (ours)	5.40 ± 0.07	24.28 ± 0.11
	BC+ (ours)	$\mathbf{5.22 \pm 0.04}$	$\mathbf{23.68 \pm 0.10}$
ResNet-29† [28]	Standard	4.24 ± 0.06 / 4.39 [28]	20.18 ± 0.07
	BC (ours)	3.75 ± 0.04	19.56 ± 0.10
	BC+ (ours)	$\mathbf{3.55 \pm 0.03}$	$\mathbf{19.41 \pm 0.07}$
ResNeXt-29 (16×64d)† [28]	Standard	3.54 ± 0.04 / 3.58 [28]	$\mathbf{16.99 \pm 0.06}$ / 17.31 [28]
	BC (ours)	$\mathbf{2.79 \pm 0.06}$	18.21 ± 0.12
	BC+ (ours)	2.81 ± 0.06	17.93 ± 0.09
DenseNet-BC ($k = 40$)† [13]	Standard	3.61 ± 0.10 / 3.46 [13]	17.28 ± 0.12 / 17.18 [13]
	BC (ours)	2.68 ± 0.03	16.36 ± 0.10
	BC+ (ours)	$\mathbf{2.57 \pm 0.06}$	$\mathbf{16.23 \pm 0.07}$
Shake-Shake Regularization [9]	Standard	2.86 [9]	**15.85** [9]
	BC (ours)	2.38 ± 0.04	15.90 ± 0.06
	BC+ (ours)	$\mathbf{2.26 \pm 0.01}$	16.00 ± 0.10

epochs training, we also started training with a learning rate of 0.1 and then divided the learning rate by 10 at the epoch in $\{50, 90, 120, 140\}$. We reported classification errors on the validation set using both single-crop testing [28] and 10-crop testing [17].

The results are shown in Table 1. We also show the training curves in Fig. 4. The performance of BC learning with the default 100-epochs training was significantly improved from that of standard learning. Moreover, the performance of BC learning was further improved with 150-epochs training, while that of standard learning was not improved, and we achieved 19.43%/4.80% single-crop top-1/top-5 validation errors and 18.22%/4.13% 10-crop validation top1/top5 errors. The single-crop top-1 error was improved by around 1% compared to the default performance reported in [28].

Discussion. Learning between-class examples among 1,000 classes is difficult, and it tends to require a large number of training epochs. As shown in Fig. 4, the performance on the first 100 epochs of the 150-epochs training of BC learning is worse than the performance of standard learning. Therefore, the learning schedule should be carefully designed. Furthermore, we assume that the usage of curriculum learning [4] would be helpful to speed up the training; namely, at the early stage, we generate a mixing ratio close to 0 or 1 and input relatively pure examples, and we gradually change the distribution of r to flat.

4.2. Experiments on CIFAR

Now, we compare the performance of standard and BC learning on CIFAR-10 and CIFAR-100 datasets [16]. We trained the standard 11-layer CNN, ResNet-29 [28], ResNeXt-29 (16×64d) [28], DenseNet (BC, $k = 40$) [13], and Shake-Shake Regularization (S-S-I and S-E-I for CIFAR-10 and CIFAR-100, respectively) [9]. To validate the comparison, we also incorporated BC learning into their original Torch [6] training codes[4]. The 11-layer CNN was also incorporated into one of them. All of these codes use the standard shifting/mirroring data augmentation scheme that is widely used for these two datasets. We added 75 training epochs with a further smaller learning rate (1/10) to the default learning schedule of a total of 300 epochs for ResNet-29, ResNeXt-29, and DenseNet. We show the difference from the default settings in the appendix, as well as the configuration of 11-layer CNN. We trained each model 10 times for the 11-layer CNN and ResNet-29, and 5 times for other networks. We report the average and the standard error of the final top-1 errors.

We summarize the results in Table 2. The performances of all networks on CIFAR-10 were improved with the simple BC learning. Furthermore, with the improved version of BC learning (BC+), which treat the image data as waveforms, the performance was further improved. The best result on CIFAR-10 was 2.26% on Shake-Shake Regularization. The performance was stable, and the error rate of all 5 trials were in the range of 2.25%–2.28%. We do not know whether this result should be counted as the state-of-the-art; however, BC learning proves to be able to improve the performance of various networks, from a simple network to the state-of-the-art networks.

[4] https://github.com/facebookresearch/ResNeXt, https://github.com/liuzhuang13/DenseNet, https://github.com/xgastaldi/shake-shake

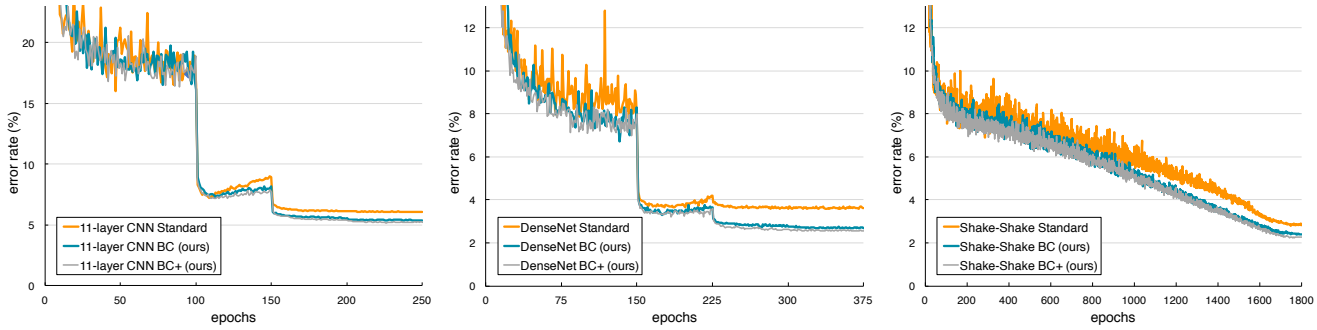


Figure 5. Training curves on CIFAR-10 (average of all trials).

We also show the training curves in Fig. 5. Note that the training curves represent the average of all trials. Contrary to the training curves on ImageNet-1K, the testing error of BC learning decreases at almost the same speed as the standard learning in the early stage of the training. Furthermore, the last 75 training epochs for 11-layer CNN and DenseNet leads to a lower testing error when using BC learning.

The performances on CIFAR-100 were also improved with BC learning. Although it may be difficult to learn the between-class examples among 100 classes with no improvement to performance on ResNeXt-29 and Shake-Shake Regularization, BC learning shows a significant improvement on 11-layer CNN, ResNet-29, and DenseNet.

Relationship with data augmentation. Here, we show the performance when using no data augmentation in Table 3. We show the average of 10 trials. As shown in this table, the degree of improvement in the performance is at the same level as, or even smaller than when using the standard data augmentation, although the variation of training data increases from 50,000 to approximately ${}_{50{,}000}C_2$. We assume this is because the potential within-class variance is small when using no data augmentation. If the within-class variance of the feature space is small, the variance of the features of the mixed images also becomes small, and the overlap in Fig. 2(top) becomes small. Therefore, the effect of BC learning becomes small as a result. We assume that BC learning is compatible with, or even strengthened by, a strong data augmentation scheme.

Table 3. Comparison when using no data augmentation. The figures between brackets indicate the error rates when using the standard data augmentation.

Model	Learning	Error rate (%) on CIFAR-10	Error rate (%) on CIFAR-100
11-layer CNN	Standard	9.68 (6.07)	33.04 (26.68)
	BC (ours)	8.38 (5.40)	31.00 (24.28)
ResNet-29 [28]	Standard	8.38 (4.24)	31.36 (20.18)
	BC (ours)	7.69 (3.75)	30.79 (19.56)

4.3. Ablation analysis

To understand the part that is important for BC learning, we conducted an ablation analysis following [3]. We trained an 11-layer CNN on CIFAR-10 and CIFAR-100 using various settings. We implemented the codes of ablation analysis using Chainer v1.24 [27]. All results are shown in Table 4, as well as the average of 10 trials.

Zero-mean & mixing method. The differences of the improved BC learning (Eqn. 4) from the simplest BC learning (Eqn. 2) are as follows: 1) we subtract par-image mean; 2) we divide the mixed image by $\sqrt{r^2 + (1-r)^2}$ considering that waveform energy is proportional to the square of the amplitude; and 3) we take the difference of image energies into consideration. We investigated which of them has a great effect. As a result, considering the difference of image energies proved to be of little significance, comparing True $\times$ Eqn. 3 and True $\times$ Eqn. 4. This would be because the variance of image energies is smaller than that of sound energies. However, per-image mean subtraction and dividing by the square root are important (True $\times$ Eqn. 4 *vs.* False $\times$ Eqn. 4 and True $\times$ Eqn. 2 *vs.* True $\times$ Eqn. 3). This result shows that treating the image data as waveforms contributes to the improvement in performance.

Label. We compared the different labels that we applied to the mixed image. The performance worsened when we used a single label ($\mathbf{t} = \mathbf{t}_1$ if $r > 0.5$, otherwise $\mathbf{t} = \mathbf{t}_2$) and softmax cross entropy loss. It would be inappropriate to train the model to recognize a mixed image as particular class. Using multi label ($\mathbf{t} = \mathbf{t}_1 + \mathbf{t}_2$) and sigmoid cross entropy loss marginally improved the performance, but the proposed ratio label and KL loss performed the best. The model can learn the between-class examples more efficiently when using our ratio label.

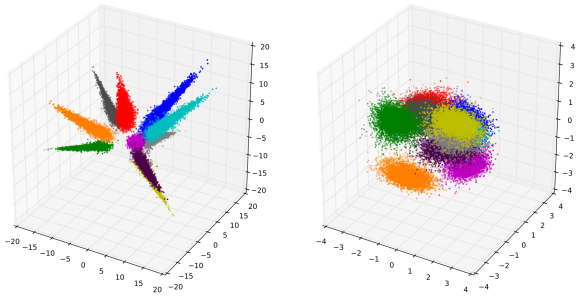
Number of mixed classes. We investigated the relationship between the performance and the number of classes of images that we mixed. Surprisingly, the performance

Table 4. Ablation analysis. We trained 11-layer CNN on CIFAR-10 and CIFAR-100 using various settings. We report the average error rate of 10 trials.

Comparison of	Setting	Error rate (%) on C-10	C-100
Zero-mean & mixing method	False × Eqn. 2 (BC)	5.40	24.28
	True × Eqn. 2	5.45	24.25
	True × Eqn. 3	**5.17**	23.72
	True × Eqn. 4 (BC+)	5.22	**23.68**
	False × Eqn. 4	5.26	23.98
Label	Single	6.35	27.28
	Multi	6.05	26.31
	Ratio (proposed)	**5.22**	**23.68**
# mixed classes	$N = 1$	5.98	26.01
	$N = 1$ or 2	5.31	23.79
	$N = 2$ (proposed)	5.22	**23.68**
	$N = 2$ or 3	**5.15**	23.78
	$N = 3$	5.32	24.20
Where to mix	Input (proposed)	**5.40**	24.28
	pool1	5.74	**24.09**
	pool2	6.52	25.38
	pool3	6.05	27.40
	fc4	6.05	26.70
	fc5	6.12	25.99
Standard learning		6.07	26.68

was improved when we mixed two images belonging to the same class ($N = 1$). Additionally, if we selected two images completely randomly and allowed the two images to be sometimes the same class ($N = 1$ or 2), the performance was worse than proposed $N = 2$, in which two images were always different classes. Because BC learning is a method of providing a constraint to the feature distribution of different classes, we should select two images belonging to the different classes. We also tried to use mixtures of three different classes with a probability of 0.5 in addition to the mixtures of two different classes ($N = 2$ or 3), but the performance was not significantly improved from $N = 2$. Moreover, the performance when we used only the mixtures of three different classes ($N = 3$) was worse than that of $N = 2$ despite the larger variation in training data. We assume that mixing more than two classes cannot efficiently provide a constraint to the feature distribution.

Where to mix. We investigated what occurs when we mix two examples within the network. Here, we used the simple mixing method of Eqn. 2. The performance was also improved when we mixed two examples at the layer near the input layer. We assume this is because the activations of lower layers can be treated as waveforms because the spatial information is preserved to some extent. Additionally, mixing at the layer close to the output layer had little effect on the performance. It is interesting that the performance was worsened when we mixed two examples at the middle point of the network (pool2 and pool3 for CIFAR-10 and CIFAR-100, respectively). It is expected that the middle layer of the network extracts features that represent both spatial and semantic information simultaneously, and mixing such features would not make sense for machines.



Standard learning BC learning (ours)

Figure 6. Visualization of feature distributions using 3-D PCA. BC learning indeed imposes a constraint on the feature distribution.

4.4. Visualization

Finally, we visualize the features learned with standard and BC learning in Fig. 6. We applied PCA to the activations of the 10-th layer of the 11-layer CNN trained on CIFAR-10 against the training data. As shown in this figure, the features obtained with BC learning are spherically distributed, and have small within-class variances, whereas that obtained with standard learning are widely distributed from near to far the decision boundaries. We conducted further analysis on the learned features in the appendix. In this way, BC learning indeed imposes a constraint on the feature distribution, which cannot be achieved with standard learning. We conjecture that is why the classification performance was improved with BC learning.

5. Conclusion

We proposed a novel learning method for image classification called BC learning. We argued that CNNs have an aspect of treating input data as waveforms, and attempted to apply a similar idea to what we did for sounds. As a result, the performance was significantly improved by simply mixing two images using internal divisions and training the model to output the mixing ratio. Moreover, the performance was further improved with a mixing method that treats the images as waveforms. BC learning is a simple and powerful method that can impose a constraint on the feature distribution. We assume that BC learning can be applied not only to images and sounds but also to other modalities.

Acknowledgement

This work was supported by JST CREST Grant Number JPMJCR1403, Japan.

References

[1] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM TASLP*, 22(10):1533–1545, 2014. 1

[2] S. An, F. Boussaid, and M. Bennamoun. How can deep rectifier networks achieve linear separability and preserve distances? In *ICML*, 2015. 3

[3] Authors. Learning from between-class examples for deep sound recognition. *ICLR*, 2018. https://openreview.net/forum?id=B1Gi6LeRZ. 1, 2, 3, 4, 7

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *ICML*, 2009. 6

[5] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *TCOM*, 31(4):532–540, 1983. 4

[6] R. Collobert, S. Bengio, and J. Mariéthoz. Torch: a modular machine learning software library. Technical report, Idiap, 2002. 5, 6

[7] W. Dai, C. Dai, S. Qu, J. Li, and S. Das. Very deep convolutional neural networks for raw waveforms. In *ICASSP*, 2017. 4

[8] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936. 3, 10

[9] X. Gastaldi. Shake-shake regularization. In *ICLR Workshop*, 2017. 2, 5, 6, 10, 11

[10] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 11

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[12] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017. 1

[13] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. 5, 6, 10, 11

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 1, 11

[15] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[16] A. Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report*, 2009. 6

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 6

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *IEEE*, 1998. 1

[19] K. J. Piczak. Environmental sound classification with convolutional neural networks. In *MLSP*, 2015. 1

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 5

[21] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals. Learning the speech front-end with raw waveform cldnns. In *Interspeech*, 2015. 1, 4

[22] K. S. Shanmugam, F. M. Dickey, and J. A. Green. An optimal frequency domain filter for edge detection in digital pictures. *TPAMI*, (1):37–49, 1979. 4

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(Jun):1929–1958, 2014. 1, 11

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[26] Y. Tokozume and T. Harada. Learning environmental sounds with end-to-end convolutional neural network. In *ICASSP*, 2017. 1, 4

[27] S. Tokui, K. Oono, and S. Hido. Chainer: a next-generation open source framework for deep learning. In *NIPS Workshop on Machine Learning Systems*, 2015. 7

[28] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 1, 2, 5, 6, 7, 10, 11

[29] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 1, 4

[30] X. Zhang, Z. Li, C. C. Loy, and D. Lin. Polynet: A pursuit of structural diversity in very deep networks. *arXiv preprint arXiv:1611.05725*, 2016. 1

A. Analysis on learned features

In Fig. 6 of the main paper, we visualized the learned features of 11-layer CNN and demonstrated that BC learning has the ability to impose a constraint on the feature distribution. Here, we show the results of more detailed analysis on the learned features. We used the same model as that used in Fig. 6 of the main paper.

A.1. Fisher's criterion

We calculated Fisher's criterion [8] for all combinations of two classes. Let $\{\mathbf{x}_n\}_{n \in C_i}$ and $\mathbf{m}_i$ be features of class C_i and the average of them ($\frac{1}{N_i} \Sigma_{n \in C_i} \mathbf{x}_n$), respectively. Here, Fisher's criterion between class C_1 and C_2 is defined as:

$$\frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}}, \tag{5}$$

where:

$$\begin{aligned} \mathbf{S}_B &= (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^\top, \\ \mathbf{S}_W &= \Sigma_{n \in C_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^\top \\ &\quad + \Sigma_{n \in C_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^\top, \\ \mathbf{w} &\propto \mathbf{S}_W^{-1} (\mathbf{m}_1 - \mathbf{m}_2). \end{aligned} \tag{6}$$

We show the mean Fisher's criterion in Table 5. We used the activations of the 10-th layer against training data. Fisher's criterion of the features learned with BC learning was larger than that of standard learning. This result shows that a discriminative feature space is learned with BC learning.

Table 5. Comparison of mean Fisher's criterion. BC learning indeed enlarges Fisher's criterion in the feature space.

Learning	Mean Fisher's criterion
Standard	1.76
BC (ours)	**1.97**

A.2. Activation of final layer

We investigated the activation of the final layer against training images. The results are shown in Fig. 7. The (i, j) element represents the mean activation of i-th neuron of the final layer (before the softmax) against the training images of class j. As shown in this figure, each neuron responds only to the corresponding class when using BC learning. The responses against other classes are almost the same level and most of them are negative values, whereas the responses against classes other than the corresponding class have a large variance and some of them are positive values when using standard learning. This result indicates that the features of each class learned with BC learning are distributed in the opposite side of the features of other classes. Such features can be easily separated. It is possible that BC learning indeed regularizes the positional relationship among the feature distributions.

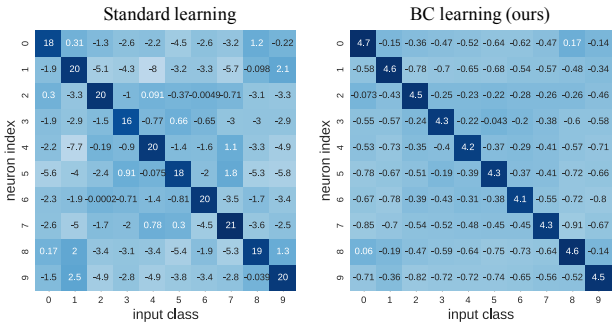


Figure 7. Activation of the final layer. The (i, j) element represents the mean activation of i-th neuron of the final layer (before the softmax) against class j.

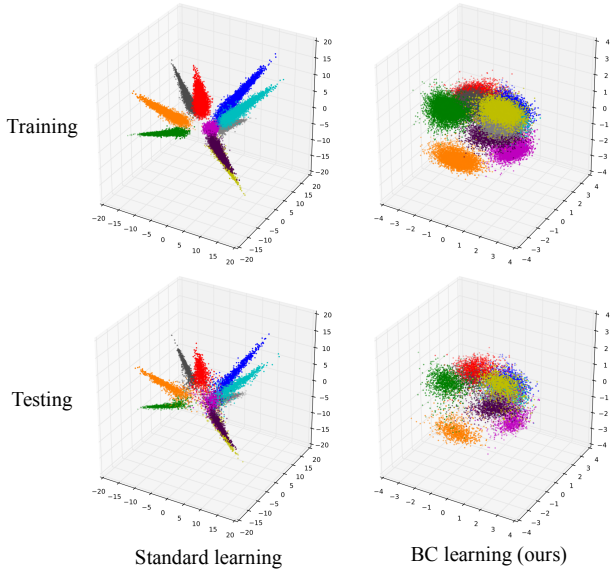


Figure 8. Feature distributions against training and testing data.

A.3. Training features *vs.* testing features

We compared the feature distributions against training and testing data. We visualized the activations of the 10-th layer using 3-D PCA. The results are shown in Fig. 8. The feature distributions of BC learning against training and testing data have similar shapes, whereas some testing examples are projected onto the points near the origin when using standard learning. This result indicates that the model learned with BC learning has a higher generalization ability.

B. Details of CIFAR experiments

B.1. Learning settings

We summarize the learning settings for CIFAR experiments in Table 6. Although most of them follow the original learning settings in [28, 13, 9], we slightly modified the learning settings for ResNet-29, ResNeXt-29 [28], and

Table 6. Learning settings for CIFAR experiments. The figures between brackets indicate the default learning settings.

Model	# of epochs	Initial LR	LR schedule	nGPU	mini-batch size
11-layer CNN	250	0.1	{100, 150, 200}	1	128
ResNet-29 [28]	**375** (300)	**0.0125** (0.1)	{**150**, **225**, **300**} ({150, 225})	**1** (8)	**16** (128)
ResNeXt-29 [28]	**375** (300)	0.1	{**150**, **225**, **300**} ({150, 225})	8	128
DenseNet [13]	**375** (300)	0.1	{**150**, **225**, **300**} ({150, 225})	4	64
Shake-Shake [9] (CIFAR-10)	1800	0.2	cosine	2	128
Shake-Shake [9] (CIFAR-100)	1800	0.025	cosine	2	32

DenseNet [13] in order to achieve a satisfactory performance. We trained the model by beginning with a learning rate of *Initial LR*, and then divided the learning rate by 10 at the epoch listed in *LR schedule*, except that a cosine learning rate scheduling was used for Shake-Shake. We then terminated training after *# of epochs* epochs.

B.2. Configuration of 11-layer CNN

We show the configuration of 11-layer CNN in Table 7. We applied ReLU activation for all hidden layers and batch normalization [14] to the output of all convolutional layers. We also applied 0.5 of dropout [24] to the output of fc4 and fc5. We used a weight initialization of [10] for all convolutional layers. We initialized the weights of each fully connected layer using the uniform distribution $U(-\sqrt{1/n}, \sqrt{1/n})$, where n is the input dimension of the layer. By using BC learning, we can achieve around 5.2% and 23.7% error rates on CIFAR-10 and CIFAR-100, respectively, despite the simple architecture.

Table 7. Configuration of 11-layer CNN.

Layer	ksize	stride	pad	# filters	Data shape
Input					(3, 32, 32)
conv1-1	3	1	1	64	
conv1-2	3	1	1	64	
pool1	2	2			(64, 16, 16)
conv2-1	3	1	1	128	
conv2-2	3	1	1	128	
pool2	2	2			(128, 8, 8)
conv3-1	3	1	1	256	
conv3-2	3	1	1	256	
conv3-3	3	1	1	256	
conv3-4	3	1	1	256	
pool3	2	2			(256, 4, 4)
fc4				1024	(1024,)
fc5				1024	(1024,)
fc6				# classes	(# classes,)