Between-Class Learning for Image Classification (ECE57000 Fall 2022)

Kenneth Wong Hon Nam
wong330@purdue.edu
Purdue University USA

Abstract

This paper will be a detailed discussion of the extension of a paper submitted to the 2018 CVPR conference (Tokozume, 2018), discussing between-class learning for image classification. This paper will include the motivation behind the experiment, related works, implementation, results, evaluation of results the researcher has managed to obtain, and significant discussion/ problems that the researcher has come across.

1. Motivation

Image Classification is a big part of Machine Learning and there exists a lot of datasets of images readily available for people to train their classification model. Between-Class is a novel way of training Image-Classification models.

The paper took inspiration from digital sound mixing (Tokozume et al., 2017). Mixing sounds makes sense to humans as mixing sounds is a superimposition of multiple sound waves, hence we can distinguish between the different sound waves. However, mixing images does not give meaning to humans, mixing pixel values together does not visually make sense. Therefore, the researchers argued that perhaps machines can perceive and abstract useful information that humans can't perceive from a mixed image.

The researchers mixed two images from two different classes with a naive (BC) and advanced (BC+) algorithm. They then trained the model to output the mixing ratios between the two classes. The BC+ algorithm is motivated by sound mixing, treating image pixel intensities as sound energies.

The paper (Tokozume, 2018) argues that BC learning has the ability to impose constraints on the shape of the feature distribution, improving the model's ability to generalize.

. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Related work

2.1. Between class learning for image classification

2.1.1. Summary

This paper was accepted to the 2018 CVPR conference and discusses their novel machine learning method called Between-Class learning. The main focus of their method was on image classification, using image-classification models to test the accuracy and performance of their method.

With this, the researchers tried two methods of mixing, one slightly more complicated than the other. The researchers found that their Between-Class machine learning method achieved 19.4% and 2.26% top-1 errors on ImageNet-1K and CIFAR-10 respectively (Tokozume, 2018). This is important as it shows that we can further generate more meaningful test data to better train our models.

The main motivation of their paper was the idea of how humans can identify different sounds of different pitches despite them being superimposed on one another. The researchers thought that if we can abstract information from what seems like nonsense if we fed the sound to a machine, they thought that mixing two images together, which is meaningless data to us, may yield results in a machine as they may be able to abstract the two images from one another.

Similar to the concept of sound, the researchers treat the input image as a sound wave and mix two images together, and instead of having the model predict the image, the researchers want to model to predict the mix ratio of the two images.

2.1.2. Critical Review: Strengths

The strengths of the paper were that it cited and used a lot of related work. They based their between-class model on a paper (Dai, 2017) that figured out the mathematics of converting waveforms to an input vector by incorporating the "energy" of the waveform. This gives the paper a lot of credibility as they worked on a previously accepted mathematical operation to transform their image inputs.

Furthermore, the paper shows a lot of statistics with community-accepted image classification datasets and models. The researchers used large datasets such as ImageNet-1K, CIFAR-100, and CIFAR-10. The researchers used large image classification models such as ResNeXt-29, ResNet-29, DenseNet, and Shake-Shake Regularization to make sure that their Between-Class learning method worked.

Lastly, I believe that it was interesting that the researchers included comparisons between the BC and BC+ models, showing us how a slight optimization made the error rate decrease.

2.1.3. Critical Review: Weakness

The weakness associated with this paper is that the models have not gone through a large number of epochs or trials to be determined consistently. The highest epoch that the researchers conducted was on the ImageNet-1K, having 150 epochs, which is a rather small amount of epochs considering it is an enormous dataset. Furthermore, I thought that their way of mixing the labels and finding the ratio was unconvincing as it appeared to only be limited to classifying two classes at a time due to the ratio, which is not applicable to large datasets with 1000+ classes.

Lastly, the researchers claimed to have imposed a constraint on the feature distribution by simply giving us a 3D Data Visualization of the data points after PCA dimensionality reduction without giving us information about the variance, mean, etc...

2.2. Human uncertainty makes classification more robust

2.2.1. Summary

This paper was accepted to the CVPR 2019 conference. This paper explores the effectiveness and accuracy of adding human uncertainty into an image classification dataset. The researchers modified a popular community-accepted dataset CIFAR-10 and added human uncertainty soft labels to the dataset, calling it CIFAR10H. The CIFAR10H dataset itself consists of 10,000 images with 511,400 human categorization decisions (Peterson, 2019).

The researchers discovered that modifying their dataset to be soft-labeled with human uncertainty instead of one hot encoding, it has shown the following improvements to model training:

- The dataset improved the generalizability of the model as compared to if the model was trained with hard labels.
- lower softmax errors whenever the model makes an incorrect prediction of an image class.
- Model was significantly more resistant to adversarial attacks.

2.2.2. Critical review: Strength

The strengths of the paper were that the researchers were very comprehensive. they utilized 5 image classification datasets (CIFAR-10, CIFAR-10 .1v6, v4, CINIC10, ImageNet-Far) other than their novel CIFAR10H. This makes their research a lot more reliable and factual as it compared their results with widely accepted datasets.

Moreover, the researchers provided a lot of informative graphs, tables, and statistics to back up their claims on the training accuracy and error of the training results of their paper. I also really appreciated how the researchers delved into the topic of whether or not the model was learning the human uncertainty when given the modified dataset or if the model still utilizes the information that is present in the image.

Lastly, a final strength of the paper was that they detailed every step of the making of their dataset, breaking down the costs, manpower, and time taken to create the dataset. Moreover, the dataset itself is rich in information, having more than half a million human categorization decisions.

I believe that this paper does discover a novel idea of dataset image labeling as their results do show promise. By training ImageNet-Far with CIFAR10H, its cross-entropy error rate was reduced by 38% on average as compared to training the model with the original CIFAR-10 (Peterson, 2019).

2.2.3. Critical Review: Weakness

A weakness of this paper is that the researchers did not fully back up their claim on the robustness of adversarial attacks. The researchers did provide a graph showing that the cross-entropy loss of soft labels is a lot less than hard labels during an adversarial attack, but that is the only evidence, that I felt was a bit lackluster.

Another point that I would like to point out is that the researchers brought in a lot of calculations and assumptions without elaboration on how or where these methods came to be or were derived from.

2.3. Class-Balanced Loss Based on Effective Number of Samples

2.3.1. Summary

This paper was accepted to the 2019 CVPR conference and discusses their solution to solve loss based on class imbalances in image classification datasets. The issue with

long-tailed datasets is that there is an imbalance of training data for different classes, resulting in poor predictions for classes with a smaller sample of training data. Moreover, there also is a diminishing return to classes with a large sample of training data as it leads to waste in computation and issues with overfitting.

Numerous studies (Bengio, 2015; Ouyang, 2016; Huang, 2016) have tried to solve this issue through cost-sensitive re-weighting, which is to set a hyper-parameter on the loss function based on the ratio between the number of training data for a certain class and the total size of the training set. However, the paper argues that this is not generalizable and in some cases hurts the model by overfitting.

The researchers designed an effective re-weighting scheme that uses the effective samples for each class to re-balance the loss to calculate a class-balanced loss. (Yin Ciu, 2019) They trained their model on artificial and naturally long-tailed datasets on CNN models and found that the network is able to achieve significant performance gains as compared to without class re-balancing.

2.3.2. Critical Review: Strength

What I believed was a big strength of this paper was the number of references it had to other papers that tackled a similar problem. The researchers found similar results and referenced those papers. The paper also explained their motivation and methods very clearly, every step was very concrete and had rich references as to why they chose to set up their equations the way they are without bombarding us with unappetizing mathematics.

In terms of their experiment, the researchers used a wide range of widely popular image classification datasets such as iNaturalist 2017, 2018, and ImageNet data (ILSVRC 2012). The researchers also modified CIFAR-10 and CIFAR100 to skew the image distributions such that it was tailed to mimic real-world data. The wide range of datasets used in the experiment debiases the experiment and prevents the conclusion that the results are only applicable to a specific dataset.

2.3.3. Critical Review: Weakness

The weaknesses of this paper would be the conclusion of their experiment. I believe that the paper set up a great theory of their class-based solution and came up with great solutions such as the modification of focal loss, sigmoid loss, and cross-entropy loss. But the results of the experiment showed little to no data backing up that their experiment worked. The researchers showed a few graphs with the error rate of each model with a specific class-based loss function. However, the researchers failed to elaborate what the conditions were in these experiments, lacking information such as epochs, optimizers, folds etc...

Not a weakness but rather an observation would be that the researchers conclude by declaiming their original claim that ratio-based loss functions are inefficient, contradicting it. This is because they found that if they set their hyper-parameter to 0.99, that is when the class-based re-balancer works the best. However, 0.99 is close, if not the same as the ratio-based rebalancing method.

3. Implementation

3.1. Experiment

In the beginning, I thought of re-implementing the experiment, but I thought that that would yield no meaning as they already have the results. Hence I thought of extending the BC-Learning method. So instead of having the model guess the ratio, I would train the model to identify the class of image with the larger mix ratio. So this simplifies the experiment to an image classification task but with an between-class image mixing algorithm.

For this experiment I will be testing Resnet-18, Convnet and Shake-Shake regularization over the CIFAR-10 dataset, which has been augmented by either the BC or BC+ mixing algorithm. The reason for choosing these models is because it was chosen to be used in the experiment and that it is also a widely popular image-classification model. Furthermore, Resnet-18 will be pre-trained, Convnet and Shake-Shake Regularization would be built ourselves with the criterion outlined in the experiment.

The original experiment trains over various datasets such as CIFAR-100 and ImageNet-1K. However, since I have limited resources, I will only use the CIFAR-10 dataset.

In the following section I will be going over the mixing algorithms that I implemented. The data augmentation to the CIFAR-10 image, details of the experiment, the training process and results.

3.2. Mixing algorithm

3.2.1. BC Simple mixing

The first method of mixing is the BC mixing method, which is the naive mixing method that was discussed in the paper. We first preprocess the two images. x_1 and x_2 are pre-processed images of different classes, having the same size as the input size of the network. We then generate a mixing ratio r from a normal distribution $U(0.5, 1)$. I chose the lower bound of the normal distribution to be 0.5 as we want the class of the larger of the two ratios of images to be the label of the image. We then mix the two images with r by

the equation:

$$r_1 + (1 - r) \cdot (x_2) \tag{1}$$

The researchers regarded this equation as being the naive equation as they were trying to apply the principals of sound wave superposition to images. 0 in a soundwave is the absolute center, and any distance from 0 represents the sound energy. In this equation there is no incorporation of the idea of absolute center, nor the concept of "energy".

3.2.2. BC+ MIXING

The second method of mixing is the BC+ mixing method, which is the mixing method that incorporates elements of sound energy. Here we rewrite image as a wave equation with a static and wave component $x_i = \mu_i + d$, where μ_i is the mean of the image and d is the wave component of the image. With this, we manage to incorporate the idea of absolute center into each image. Hence we can now rewrite (1) as $r \cdot (\mu_1 + d_1) + (1 - r) \cdot (\mu_2 + d_2)$. By separating the static and wave components we get $\{r \cdot \mu_1 + (1-r) \cdot \mu_2\} + \{rd_1 + (1 - r)d_2\}$. The researchers opted to remove the static component as they argued that if we were to mix the images, the static component of the two wave-forms should be the same.

Hence, we remove the static component by subtracting the per-image mean, normalizing the image to be a 0 mean wave. Again from (1), we can get closer to relating images as waveforms by having an absolute center and having a wave component.

$$r \cdot (x_1 - \mu_1) + (1 - r) \cdot (x_2 - \mu_2) \tag{2}$$

Lastly we will normalize our equation by the ratio term and substitute the ratio term with a term that relates the standard deviations of the two images to further relate the two images. Hence we replace r with p. We get the final mixing equation:

$$\frac{p \cdot (x_1 - \mu_1) + (1 - p) \cdot (x_2 - \mu_2)}{\sqrt{p^2 + (1 - p)^2}}, \quad \textbf{where } p = \frac{1}{1 + \frac{\sigma_1}{\sigma_2} \cdot \frac{1-r}{r}} \tag{3}$$

3.3. Preprocessing

The preprocessing approach for the CIFAR-10 dataset follows the one proposed by the researchers. It is originally intended to be for the ImageNet-1K dataset (Tokozume, 2018). The researchers did not mention the same data augmentation process for the CIFAR-10 dataset, but after looking at their source code for their experiment (Tokozume, 2017), I found that they implemented the same image transformation as ImageNet-1K. In Figure 1, the data augmentation involves random-cropping, flipping and normalization of the image. The augmented images are then mixed with the unaugmented training set to expand our training dataset. According to (Krizhevsky et al., 2012), this image augmentation scheme approximately captures an important property of natural images and prevents overfitting. According to the paper (Krizhevsky et al., 2012), this scheme has reduced the top-1 error rate by over 1% .

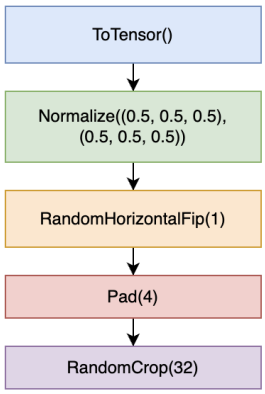

Figure 1: CIFAR-10 Preprocessing process (Krizhevsky et al., 2012)

Figure 2 is an example of the dataset after image preprocessing and BC and BC+ mixing algorithm.

4. Training

4.1. Resnet-18 (Pytorch, 2016)

For Resnet-18, I chose to use the pretrained model from Pytorch (Pytorch, 2016). In the original paper the researchers used ResNet-29 instead. I chose to use ResNet-18 as it is easily accessible from the Pytorch models module. Furthermore, 29 layers is a lot of parameters to train and I had insufficient resources to provide such training.

I made augmentations to the Resnet-18 model, by setting the fully-connected layer to the dimensions of the images in the CIFAR-10 dataset. Then I set layer 1, 2, 3, 4 and the conv1 layer to require gradient. This is to make the model to be more adaptive to the augmented dataset as the Resnet-18 model was pretrained on an unaugmented CIFAR-10 dataset.

As for the training, I trained the model over 20 epochs, with SGD optimizer, which is the optimizer used in the paper. I initially wanted to use Adam but according to a research (Keskar and Socher, 2017), it is shown that SGD outperforms Adam in training resnet neural net over CIFAR-10 and CIFAR-100.

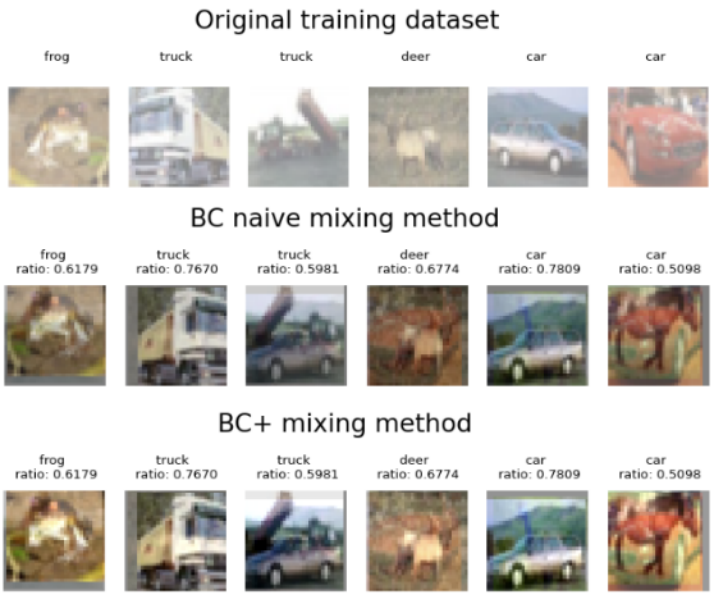

Figure 2: CIFAR-10 images after different mixing algorithms

I set the learning rate to 0.0001 and momentum to 0.8. As it is a pretrained model, the learning rate can be lower. The loss function used is Cross Entropy Loss.

4.2. Convnet

For Convnet, I followed the implementation of the paper's model, which is published on their github (Tokozume, 2017). The model has 8 convolutional layers and 3 fully-connected layers. Convnet is used as it is highly popular architecture for image-classification.

As for training, the model was trained over 20 epochs, with SGD optimizer. The learning rate is set to 0.01 with a momentum of 0.8. The loss function used is Cross Entropy Loss.

4.3. Shake-Shake Regularization (Gastaldi, 2017)

The last model used was Shake-Shake Regularization, which was regarded state-of-the-art in the writing of the original paper. For the Shake-Shake Regularization I used the model published (Viehmann, 2016).

The original Shake-Shake Regularization model used in the paper has a depth of 26. The one used in the experiment has a depth of 8 due to the lack of resources to train a deeper neural network. The epochs, loss function, learning rate and optimizer is the same as the ones of Convnet.

4.4. Custom Classes

I implemented custom classes to make the training code clean, modular and reusable to prevent rewriting the exact same code for each of the models.

First class I created was a **Mixer class**, which was responsible for loading the dataset and mixing of the dataset. This class allows users to choose their mixing algorithm and returns the training dataset along with the images after mixing. This makes it a lot simpler to load datasets for the experiment as all the logic is abstracted to a simple class interface.

The second class I implemented was the **ModelManager** class, which as the name suggests is used to handle the model training, testing and initialization. The user can input the model, optimizer and loss function and call the train or test function. By default if no loss function is provided a custom KL-divergence from (Tokozume, 2017) is used. However, I strongly suggest that the user input a loss function as the KL-divergence loss function is unstable, the reasons for this is explained in the discussion section.

The last class that I implemented was the **DataIterator** class, which concats the images and labels such that it can be inputted into the Pytorch DataLoader class for training and testing.

5. Results

The results of the experiment are interesting as we managed to get the opposite results of the original paper, hav-

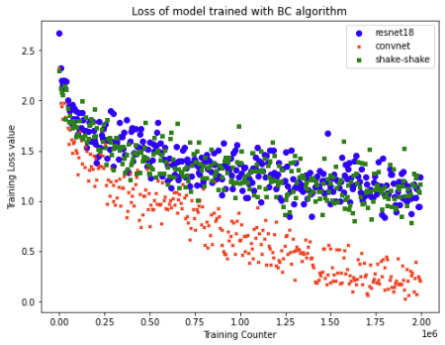

Figure 3: BC algorithm training loss

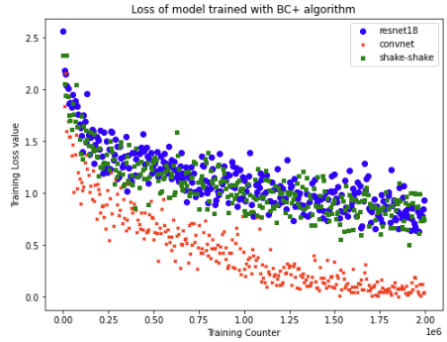

Figure 4: BC+ algorithm training loss

ing the BC mixing algorithm perform better than BC+ the mixing algorithm. From Table 1, we see that in both ResNet-18 and Shake-Shake Reg the test accuracy when the model is trained on the original dataset has the highest accuracy, which is as expected as that is the closest to the testing dataset. Shake Shake Regularization model scored the highest accuracy, which was what I expected as this model was the state-of-the-art image classification model when the original paper was written. However, when Convnet is trained on the training set with the BC mixing algorithm has the highest accuracy.

The surprising results were that the BC+ mixing model scored significantly lower than any of the models. The accuracy drop is significant such that it appears to be untrained. In ResNet-18 the drop in accuracy was 9%, which was high but not an enormous change. But the drop in accuracy in Convnet and Shake Shake is great. Accuracy dropped by 43% in Convnet and 53% in Shake Shake Regualarization model. This is the complete opposite to the results that were expected as the original paper boasted an increase in accuracy from BC mixing to BC+ mixing. This anomaly will be discussed in the discussion and evaluation section that follows.

From Figure 3 and Figure 4, we see that when training the models with the BC+ mixing algorithm our models loss converges a lot faster than the BC mixing algorithm. Surprisingly, convnet has the lowest training loss and converged a lot faster than both resnet-18 and convnet. I expected resnet-18 to have the lowest loss and converge the fastest as it is a pretrained model. However, this slow convergence of loss could be due to the small learning rate of 0.0001 as compared to the learning rate of 0.01 of shake shake regularization and convnet.

Despite having a higher training loss, shake shake regularization still managed to perform better than both convnet and resnet-18, which could be a sign that shake shake regularization did not overfit to the training dataset.

From Figure 4, we can see that the loss of convnet is consistently very low. With our results from Table 1, this is a clear sign of overfitting of our convnet model. The general loss of both shake shake and resnet-18 is less than BC, and with their decrease in accuracy, it is certain that the model overfitted on the training data.

6. Evaluation

From the results that are displayed in the results above, it is safe to assume that the models were a lot more prone to overfitting on the CIFAR-10 data set with BC+ mixing than with BC mixing and no mixing. This could be due to the fact that in equation 2 we normalize our images to have 0 mean, lowering the variance of each image. However, this should have been taken into account for as discussed in the BC+ algorithm, we offset the loss of the image variance by replacing our mixing ratio from r to $p = \frac{1}{1+\frac{\sigma_1}{\sigma_2}\cdot\frac{1-r}{r}}$.

I believe that with the BC+ mixing algorithm, the algorithm only shows results if we train the model to output the mixing ratio between the two classes as described in the original paper (Tokozume, 2018). The paper touched upon this as by having our training label be of only one class removes a lot of information from the image as we are removing the contents of the other image $r < 0.5$. By removing the information of the other class the experiment is reduced to a data augmentation training, where the mixing of the dataset is an image transformation to increase testing data, rather than between-class learning.

Despite the argument above, it is still unusual that BC mixing algorithm outperformed standard learning in the Convnet model in Figure 1. I speculate that this is due to the perseverance of information from both images as we did not normalize the images.

7. Discussion

This is a limited extension of the original paper's experiment due to the lack of resources as I do not have access to

	Accuracy of model (%)	
Model	Learning	CIFAR-10
ResNet-18 (Pytorch, 2016)	Standard	79 ± 1
	BC	73 ± 2
	BC+	64 ± 1
Convnet (Tokozume, 2017)	Standard	81 ± 1
	BC	82 ± 1
	BC+	39 ± 1
Shake-Shake Reg. (Viehmann, 2016)	Standard	82 ± 1
	BC	79 ± 2
	BC+	26 ± 5

Table 1: Accuracy of training models with different mixing algorithms

a GPU. I only own a macbook air so training is exceptionally difficult. I bought the google colab pro subscription for training the model, but the plan only offers 100 computing units per month, which was quickly used up with my training as I trained multiple models multiple times with 20 epochs each to produce the results shown in Figure 1.

The original paper implements 5 high depth models with training datasets CIFAR-10, CIFAR-100 and ImageNet-1K on 100 and 150 epochs. This is a lot more thorough than my experiment as they train on multiple datasets, which prevents generalization of the results of BC. However, the maximum my google colab could handle was CIFAR-10.

I also think that the results of Shake-Shake regularization and Resnet-18 could have been improved if the model had more layers.

A big design decision I made was to use Cross Entropy Loss as my loss function instead of KL Divergence, which was what was proposed in the original paper. The reason for my switch was because I encountered an issue of an exploding gradient where my gradient would grow to inf and that the training output of all the models would result in NaN. I spent a very long time debugging this but never managed to fix it, so I ended up using Cross Entropy Loss, which worked and was a lot more stable.

I searched the internet for solutions for this explosion of gradient and found a solution to prevent my models from outputting NaN, which was to adding a soft-max layer to my output. But this did not work as my model never managed to converge and resulted in a hovering loss. I also tried lowering the learning rate and increasing the batchsize but this did not solve the issue.

Another issue that I found was that there seems to be a discrepancy in the training of my models in Pytorch 1.12 and Pytorch 1.13 CUDA. I point this out as I sent my jupyter notebook to my father to run on his computer which has a Nvidia GeForceX RTX 3050Ti graphics card, which is CUDA compatible. The training resulted in my models returning NaN. But whilst training on the ARM1 CPU of my MacBook running Pytorch 1.12 the models did not return NaN. I checked the torch forums and it appears to be a common issue amongst the community who has upgraded their Pytorch version to version 1.13. I wonder if this could also contribute to the issue of an exponential increase in gradient while training.

Overall the results of the experiment does correlate with the original experiment, that being that performing BC+ with a single label instead of a label of both classes is not optimal. I can not confirm if the performance with 2 labels perform better as this is not conducted in my experiment.

References

Yuji Tokozume. Between-class learning for image classification, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/papers/Tokozume_Between-Class_Learning_for_CVPR_2018_paper.pdf.

Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Learning from between-class examples for deep sound recognition, 2017. URL https://arxiv.org/abs/1711.10282.

W. Dai. Very deep convolutional neural networks for raw waveforms., 2017. URL https://arxiv.org/pdf/1610.00087.pdf.

Joshua C. Peterson. Human uncertainty makes classification more robust, 2019. URL https://openaccess.thecvf.com/content_ICCV_2019/papers/Peterson_Human_Uncertainty_Makes_Classification_More_Robust_ICCV_2019_paper.pdf.

Samy Bengio. Sharing representations for long tail computer vision problems, 2015. URL https://arxiv.org/pdf/2103.16370.pdf.

Wanli Ouyang. Factors in finetuning deep model for object detection with long-tail distribution, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/papers/Ouyang_Factors_in_Finetuning_CVPR_2016_paper.pdf.

Chen Huang. Learning deep representation for imbalanced classification., 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/papers/Huang_Learning_Deep_Representation_CVPR_2016_paper.pdf.

Tokozume. Between class learning, 2017. URL \url{https://github.com/mil-tokyo/bc_learning_image}.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks, 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Pytorch. Pytorch resnet18 documentation, 2016. URL https://pytorch.org/vision/master/models/generated/torchvision.models.resnet18.html.

Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd, 2017. URL https://arxiv.org/abs/1712.07628.

Xavier Gastaldi. Shake-shake regularization. *CoRR*, abs/1705.07485, 2017. URL http://arxiv.org/abs/1705.07485.

Thomas Viehmann. Quick shake-shake net for cifar10, 2016. URL https://notebook.community/t-vi/pytorch-tvmisc/misc/cifar10-shake-shake.