Class-Wise Difficulty-Balanced Loss for Solving Class-Imbalance

Saptarshi Sinha[0000−0002−5207−1551], Hiroki Ohashi, and Katsuyuki Nakamura[0000−0002−8074−2279]

Hitachi, Ltd. Research & Development Group, Tokyo, 185-8601 Japan
{saptarshi.sinha.hx, hiroki.ohashi.uo, katsuyuki.nakamura.xv}@hitachi.com

Abstract. Class-imbalance is one of the major challenges in real world datasets, where a few classes (called majority classes) constitute much more data samples than the rest (called minority classes). Learning deep neural networks using such datasets leads to performances that are typically biased towards the majority classes. Most of the prior works try to solve class-imbalance by assigning more weights to the minority classes in various manners (e.g., data re-sampling, cost-sensitive learning). However, we argue that the number of available training data may not be always a good clue to determine the weighting strategy because some of the minority classes might be sufficiently represented even by a small number of training data. Overweighting samples of such classes can lead to drop in the model's overall performance. We claim that the 'difficulty' of a class as perceived by the model is more important to determine the weighting. In this light, we propose a novel loss function named Class-wise Difficulty-Balanced loss, or CDB loss, which dynamically distributes weights to each sample according to the difficulty of the class that the sample belongs to. Note that the assigned weights dynamically change as the 'difficulty' for the model may change with the learning progress. Extensive experiments are conducted on both image (artificially induced class-imbalanced MNIST, long-tailed CIFAR and ImageNet-LT) and video (EGTEA) datasets. The results show that CDB loss consistently outperforms the recently proposed loss functions on class-imbalanced datasets irrespective of the data type (i.e., video or image).

1 Introduction

Since the advent of Deep Neural Networks (DNNs), we have seen significant advancement in computer vision research. One of the reasons behind this success is the wide availability of large-scale annotated image (e.g., MNIST [1], CIFAR [2], ImageNet [3]) and video (e.g., Kinetics [4], Something-Something [5], UCF [6]) datasets. But unfortunately, most of the commonly used datasets do not resemble the real world data in a number of ways. As a result, performance of state-of-the-art DNNs drop significantly in real-world use-cases. One of the major challenges in most real-world datasets is the class-imbalanced data distribution with significantly long tails, i.e., a few classes (also known as 'majority classes')

have much higher number of data samples compared to the other classes (also known as 'minority classes'). When DNNs are trained using such real-world datasets, their performance gets biased towards the majority classes, i.e., they perform highly for the majority classes and poorly for the minority classes.

Several recent works have tried to solve the problem of class-imbalanced training data. Most of the prior solutions can be fairly classified under 3 categories :- (1) Data re-sampling techniques [7–9] (2) Metric learning and knowledge transfer [10–13] (3) Cost-sensitive learning methods [14–17]. Data re-sampling techniques try to balance the number of data samples between the majority and minority classes by either over-sampling from the minority classes or under-sampling from the majority classes or using both. Generating synthetic data samples for minority classes [7, 18, 19] from given data is another re-sampling technique that tries to increase the number of minority class samples. Since the performance of a DNN depends entirely on it's ability to learn to extract useful features from data, "what training data is seen by the DNN" is a very important concern. In that context, data re-sampling strategies introduce the risks of losing important training data samples by under-sampling from majority classes and network overfitting due to over-sampling minority classes. Metric-learning [10, 11], on the other hand, aims to learn an appropriate representation function that embeds data to a feature space, where the mutual relationships among the data (e.g., similarity/dissimilarity) are preserved. It has the risk of learning a biased representation function that has learned more from the majority classes. Hence some works [12] tend to use sampling techniques with metric learning, which still faces the problems of sampling, as discussed above. Few recent researches have also tried to transfer knowledge from the majority classes to the minority classes by adding an external memory [20], which is non-trivial and expensive. Due to these concerns, the work in this paper focuses on cost-sensitive learning approaches. Cost-sensitive learning methods penalize the DNN higher for making errors on certain samples compared to others. They achieve this by assigning weights to different samples using various strategies. Typically, most prior cost-sensitive learning strategies [21, 22, 15] assume that the minority classes are always weakly represented. They ensure that the samples of the minority class get higher weights so that the DNN can be penalized more for making mistakes on the minority class samples. One such popular strategy is to distribute weights in inverse proportion to the class frequencies [21, 23]. However, certain minority classes might be fairly represented by a small amount of samples.

Fig. 1 gives an example of a class-imbalanced dataset where certain classes such as 'clean' and 'spread' are sparsely populated but can easily be learned to generalize by the classifier. Overweighting samples of such classes might lead to biasing the DNN's performance. In such situations, number of available training data per class might not be a good clue to determine sample weights.

Instead, we claim that the 'difficulty' of a class as perceived by the DNN might be a more important and helpful clue for weight assignment. The concept of 'difficulty' has been previously used by some sample-level weight assigning techniques such as focal loss [14] and GHM [17]. They reweight each sample

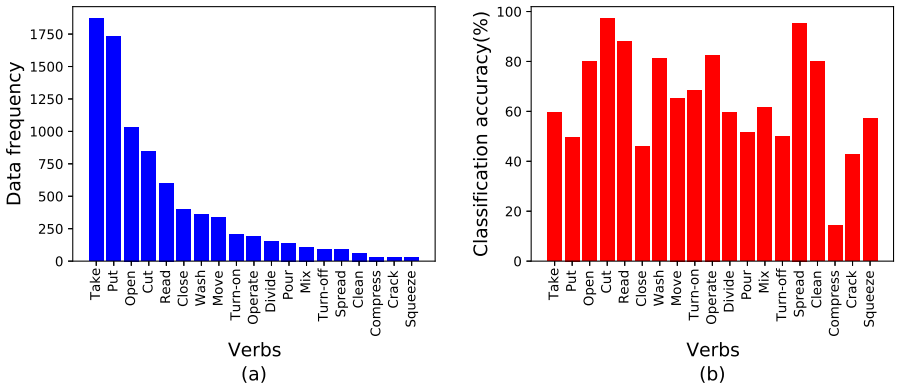

Fig. 1. (a) Class-imbalanced data distribution of EGTEA dataset. (b) Class-wise classification accuracies of a 3D-ResNeXt101 trained on the imbalanced EGTEA dataset using unweighted softmax cross-entropy loss function. It is interesting to notice that even though classes like 'Clean' and 'Spread' have very small number of data samples, the classifier finds it relatively easier to learn such classes compared to certain densely populated classes such as 'Take' and 'Put'. Therefore it is not obvious to assume that the sparsely populated classes will always be the most weakly represented.

individually by increasing weights for hard samples and reducing weights for easy samples. The increasing popularity of focal loss [14] in class-imbalanced classification tasks is based on the assumption that minority classes should have more hard samples compared to majority classes. The assumption does stand true if we compare the proportion of hard samples for the minority and majority classes. But, in terms of absolute number of hard samples, the majority classes still might surpass the minority classes simply because they have much more data samples than the minority classes. In such cases, giving high weights to all hard samples irrespective of their classes might overweight the majority classes and therefore still bias the performance of the DNN. We believe that the above drawback can be solved by considering class-level difficulty rather than sample-level. To the best of our knowledge, ours is the first work to introduce the concept of class-level difficulty for solving class-imbalance. Based on the analysis, we develop a novel weighting strategy that dynamically re-balances the loss for each sample based on the instantaneous difficulty of it's class as perceived by a DNN.

Such a strategy measures the instantaneous difficulty of each class without relying on any prior assumptions and then dynamically assigns weights to the samples of the class in proportion to the difficulty of the class. Extensive experiments on multiple datasets indicate that our class-difficulty based dynamic weighting strategy can provide a significant improvement in the performance of the commonly used loss functions under class-imbalanced situations.

The key contributions of this paper can be summarized as follows: (1) We propose a way to measure the dynamic difficulty of each class during training and use the class-wise difficulty scores to re-balance the loss for each sample, thereby giving a class-wise difficulty-balanced (CDB) loss. (2) We show that using our weighting strategy can give commonly used loss functions (e.g., cross-entropy) a significant boost in performance on multiple class-imbalanced datasets. We conduct experiments on both image and video datasets and find that our weighting strategy works well irrespective of the data type. Our research on quantifying the dynamic difficulty of the classes and using it for weight distribution might prove useful for researchers focusing on class-imbalanced datasets.

2 Related Works

As discussed in section 1, most prior works that try to solve class-imbalance can be categorized into 3 domains : (1) Data re-sampling techniques, (2) Metric learning and knowledge transfer and (3) Cost-sensitive learning methods.

2.1 Data Re-sampling

Data re-sampling techniques try to balance the number of samples among the classes by using various sampling techniques during the data pre-processing. The sampling techniques, used for the purpose, either randomly over-sample data from the minority classes or randomly under-sample data from the majority classes or both. Over-sampling from the minority classes [8, 24] replicates the available data samples in order to increase the number of samples. But such a practice introduces the risk of overfitting. Synthetic Minority Over-sampling Technique (SMOTE) [7], proposed by Chawla et al., increases the number of data samples for the minority classes by creating synthetic data using interpolation among the original data points. Though SMOTE only used the minority class samples while generating data samples, later variants of SMOTE (e.g., Borderline SMOTE [18] and Safe-level SMOTE [19]) take the majority class samples into consideration as well. But such data generation techniques do not guarantee that the synthesized data points will always follow the actual data distribution of the minority classes. On the other hand, under-sampling techniques [9, 25] reduce data from the majority classes and might result in cutting out some important data samples.

2.2 Metric Learning and Knowledge Transfer

Metric learning aims to learn an embedding function that can embed data to a feature space where the inter-data relationships are preserved. Contrastive embedding [26] is learned using paired data samples to minimize the distance between the features of same class samples while maximizing the distance between different class samples. Song et al. [10] proposed a structured feature embedding based on positive and negative samples pairs in the dataset. Triplet loss [27],

on the other hand, uses triplets instead of pairs, where one sample is considered the anchor. Metric learning still faces the risk of learning embedding functions biased towards the majority classes. Some recent works (e.g., OLTR [20]) have also tried to transfer knowledge from the majority classes to the minority classes either by meta learning [13] or by adding an external memory module [20]. Even though OLTR [20] performs well for long-tailed classification, as pointed out by [28], their design of external memory modules might be a non-trivial and expensive task.

2.3 Cost-Sensitive Learning

Cost-sensitive learning techniques try to penalize the DNN higher for making prediction mistakes on certain data samples than on the others. To achieve that, different weights are assigned to different samples and the penalty incurred on each data sample is scaled up/down using the corresponding weight. Research in this domain mainly target to find an effective way to assign these weights to the samples. To solve class-imbalance, majority of the works propose techniques that assign higher weights to the minority class samples. Such techniques ensure that the DNN gets higher penalty for making mistakes on the minority class samples. One such simple and commonly used weight distribution technique is to use the inverse class-frequencies as the weights for the samples each class [21, 23]. Later variants of this technique [22] use a smoothed version of the square root of class-frequencies for the weight distribution. Class-balanced loss [15] proposed by Lin et al. calculates the effective number of samples of each class and uses it to assign weights to the samples. All of the above mentioned works assume that the minority classes are always the most weakly represented classes and therefore needs high weights. But that assumption might not always be true because certain minority class might be sufficiently represented by a small number of samples. Giving high weights to the samples of such classes might cause drop in overall performance. Therefore, Tsung-Yi et al. proposed a sample-based weighting technique called "Focal loss" [14], where each sample is assigned a weight based on it's difficulty. The difficulty of each sample is quantified in terms of the loss incurred by the DNN on that sample, where more lossy samples imply more difficult samples. Though focal loss [14] was originally proposed for dense object detection tasks, it has also become popular in class-imbalanced classification tasks [15]. The minority classes are expected to have more difficult samples compared to the majority classes and therefore get high weights by focal loss. Indeed the proportion of difficult samples in a minority class is more than that in the majority class. However, in terms of absolute number of difficult samples, the majority class surpasses the minority class, as it is much more populated than the minority class. Therefore, giving high weights to all difficult samples irrespective of their classes still biases the DNN's performance.

Our work also lies in the regime of cost-sensitive learning. We propose a dynamic weighting system that dynamically assigns weights to each sample of each class based on the instantaneous difficulty of the class, rather than that of each sample, as perceived by the DNN. Our weighting system helps to boost

the performance of commonly used loss functions (e.g., cross-entropy loss) in class-imbalanced situations.

3 Proposed Method

3.1 Measuring Class Difficulty

Human beings use the metric 'difficulty' majorly to give a qualitative description of things, for example "this task is very difficult" or "this game is so easy". Similar behavior can also be seen in neural networks where they find some parts of a task much more difficult to perform compared to the others. For example, while training on a multi-class classification task, the classifier will find some classes easier to learn than the others. We propose to measure the difficulty of each class as perceived by the DNN and use it as clue to determine the weights for the samples. But, as difficulty is a qualitative metric, there is no direct way to add a quantitative value to it. Humans tend to classify a task as difficult, if they can not perform well in it. We use a similar approach to use the neural network's performance to measure the difficulty of classes. During training, the neural network's performance for each class is measured on a validation data set, which is then used to calculate the class-wise difficulty. The neural network's performance for any class c is measured as its classification accuracy on class c, $A_c = n_c/N_c$, where N_c denotes the total number of samples of class c in validation data and n_c denotes the number of class c samples in validation data that the model classifies correctly. Then the difficulty of class c, d_c, is measured as $d_c = 1 - A_c$. A neural network's perception of "how much a class is difficult to learn" changes as the training process of the network progresses. With time, the network's performance for each class improves and as a result, the perceived difficulty of each class also reduces. Therefore, we calculate the class-difficulties as a function of time as well. The difficulty of class c after training time (i.e., time during training) t can be calculated as

$$d_{c,t} = 1 - A_{c,t} \ , \tag{1}$$

where $A_{c,t}$ is the neural network's classification accuracy for class c on the validation data after training time t.

3.2 Difficulty-Based Weight Distribution

Once the class-wise difficulty is quantified, then it can be used to assign weights to the classes during training. It is fairly obvious that the classes, that are difficult to learn should be given higher weights compared to the easier classes. Therefore the weight for class c after training time t can be calculated as

$$w_{c,t} = (d_{c,t})^{\tau} = (1 - A_{c,t})^{\tau} \ , \tag{2}$$

where $d_{c,t}$ is the difficulty of class c after time t and τ is a hyper-parameter. The weight distribution $w_t = \{w_{1,t}, w_{2,t}, \ldots, w_{C,t}\}$ over all C classes can be computed by repeating Equation 2 for all classes.

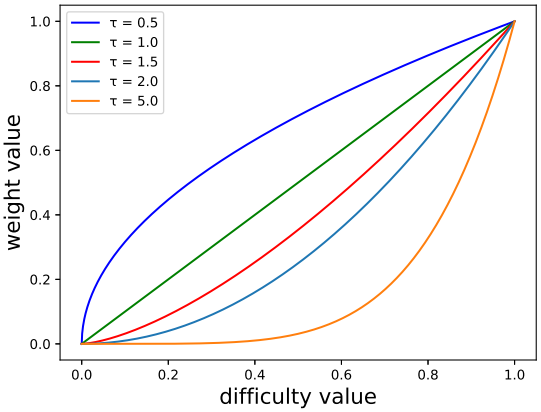

Fig. 2. Effect of changing τ on the difficulty-based weight distribution. Increasing value of τ puts heavier weights on the samples of the classes with higher difficulty, while lowering weights for the easier classes.

The hyper-parameter τ is introduced to control how much we down-weight the samples of the easy classes. Increasing value of τ relatively increases the classifier's focus on the difficult classes.

Fig. 2 shows how change in the value of τ changes the weight values for classes of different difficulties. Its effect is almost similar to that of the focusing parameter γ in focal loss [14]. The performance of our proposed method varies significantly with change in value of τ and the best value for τ differs from dataset to dataset. Unfortunately, the only way to search for the best value of τ is by trial and error. To avoid that, we propose a way to dynamically update the value of τ. For dynamically updating τ, the value of τ after training time t is calculated as

$$\tau_t = \frac{2}{1 + \exp(-b_t)} \ , \tag{3}$$

where b_t measures the bias in the performance of the classifier over C classes as

$$b_t = \frac{\max_{c=1,2,\ldots,C} A_{c,t}}{\min_{c'=1,2,\ldots,C} A_{c',t} + \epsilon} - 1 \ . \tag{4}$$

In Equation 4, ϵ is a small positive value ($= +0.0001$) introduced to handle situations where $\min_{c'=1,2,\ldots,C} A_{c',t} = 0$. Equation 3 increases the value of τ when the classification performance of the classifier is highly biased (i.e., high b_t) and decreases it in case of low bias (i.e., less b_t).

3.3 Class-Wise Difficulty-Balanced Softmax Cross-Entropy Loss

Suppose when an input data is fed to the classifier after training time t during training, the predicted output of the classifier for all C classes are $z_t =$

$\{z_{1,t}, z_{2,t}, \ldots, z_{C,t}\}$. The probability distribution $p_t = \{p_{1,t}, p_{2,t}, ..., p_{C,t}\}$ over all the classes is computed using the softmax function, which is

$$p_{j,t} = \frac{\exp z_{j,t}}{\sum_{i=1}^{C} \exp z_{i,t}} \ \forall j \in 1, 2, \ldots, C \ . \tag{5}$$

For an input data sample of class k, cross-entropy (CE) loss function computes the loss after training time t as

$$\text{CE}(p_t, k) = -\log p_{k,t} \ . \tag{6}$$

For the same input data sample, our class-wise difficulty-balanced softmax cross-entropy (CDB-CE) loss function computes the loss after training time t as

$$\text{CDB-CE}(w_t, p_t, k) = -w_{k,t} \log p_{k,t} \ . \tag{7}$$

To make the weights time-dependent, we calculate them after each epoch using the model's class-wise validation accuracy.

4 Experiments

To demonstrate our proposed solution's ability to generalize to any data-type or dataset, we evaluate the effectiveness of our solution on 4 different datasets namely MNIST, long-tailed CIFAR, ImageNet-LT and EGTEA. MNIST, long-tailed CIFAR and ImageNet-LT are image datasets while EGTEA is a video dataset.

4.1 Datasets

MNIST. From the standard MNIST handwritten digit recognition dataset [1], we generate a class-imbalanced binary classification task using a subset of the dataset. The experimental setup is exactly same as given in [29]. We select a total of 5000 training images of class '4' and '9' where '9' is chosen as the majority class. We calculate the 'majority class ratio' as

$$\text{majority class ratio} = \frac{\text{no. of training samples in majority class}}{\text{no. of training samples}} \ . \tag{8}$$

Increasing the majority class ratio increases the imbalance in the training dataset. We also use a validation set which is created by selecting 500 images for each of the two classes from the original dataset but these images are different from the 5000 images selected for training. A test set was also created by randomly selecting 800 images for each of the classes from the original MNIST test set.

Long-Tailed CIFAR. We conduct experiments on long-tailed CIFAR-100 [2]. First a validation set was created from the original training set by randomly selecting 50 images per class. After the separation of the validation set, the remaining images in the training set were used to create a long-tailed version

of the dataset using the exact same procedure as stated in [15]. The number of training images per class are reduced following an exponential function $n = n_c\mu^c$, where c is the 0-based index of the class and n_c is the remaining number of training images of class c after separation of the validation set and $\mu \in (0, 1)$. Similar to [15], the 'imbalance' factor of a dataset is defined as the number of training samples in the largest class divided by that of the smallest class. The test set used for experiment is exactly same as the original CIFAR test set available and is a balanced set.

ImageNet-LT. We also conduct experiments on the long-tailed version of the original ImageNet-2012 [3], as constructed in [20]. It comprises of 115,800 images from 1000 categories, where the most frequent class has 1280 image samples and the least frequent class has only 5 images. The test set is balanced. The split constructed by [20] also provides a validation set that is separate from the test set and training set.

EGTEA. We also conduct experiments on the EGTEA Gaze+ dataset [30]. This is an egocentric dataset that contains trimmed video clips of many kitchen-related actions. These video clips are extracted segments from longer videos that were collected by 32 different subjects. Each video clip is assigned a single action label and the challenge is to train a classifier to classify the actions from the provided video clips. Each action label is made up of a verb and a set of nouns (e.g., the action 'Wash Plate' is made up of the verb 'Wash' and noun 'Plate'). The noun classification task is very similar to the image classification task. So, for our experiments, we focus on the verb classification in order to test our proposed method on diverse tasks. This dataset has 19 different verb classes (e.g., 'Open', 'Wash' etc.). EGTEA Gaze+ [30] is inherently class-imbalanced. Fig. 1(a) shows the data distribution of the EGTEA dataset. For our experiments, we use the split1 of the EGTEA dataset (8299 training video clips and 2022 testing video clips). We create our validation set by using the training video clips from subjects P20 to P26, resulting in 1927 validation video clips and 6372 training clips.

4.2 Implementation Details

We use a LeNet-5 [31] model for MNIST unbalanced binary classification experiments following [29]. The model is trained for 4000 epochs on a single NVIDIA GeForce GTX 1080 GPU with a batch size of 100. As optimizer, we use Stochastic Gradient Descent (SGD) with momentum of 0.9, weight decay of 0.0005 and an initial learning rate of 0.001, which is decayed by 0.1 after 3000 epochs. The trained model is tested on the balanced test set.

For experiments on long-tailed CIFAR, we follow the exact same implementation strategy as provided in [15]. We train a ResNet-32 [32] model for 200 epochs using a batch size of 128 on 4 NVIDIA Titan X GPUs. We use SGD optimizer with momentum 0.9 and weight decay of 0.0005. An initial learning rate of 0.1 is used, which is decayed by 0.01 after 160 and 180 epochs.

For experiments on ImageNet-LT, we use the same setup as in [20]. We use ResNet-10 [32] model for the purpose. The trained model is tested on the balanced test data.

For EGTEA dataset, we use a 3D-ResNeXt101 [33, 34] model and train it for 100 epochs on 8 NVIDIA Titan X GPUs using a batch size of 32. SGD with momentum 0.9 is used with a weight decay of 0.0005 and an initial learning rate of 0.001, which is decayed by 0.1 after 60 epochs. During training, we sample 10 RGB frames from each video clip by dividing the clip into 10 equal segments followed by randomly selecting one RGB frame from each segment. We use random-cropping, random-rotating and horizontal-flipping as data augmentation. Training input size is $10 \times 3 \times 224 \times 224$. During testing and validation, we sample 10 RGB frames at equal intervals from each video clip.

We use PyTorch [35] framework for all our implementations. For all datasets, our CDB-CE loss implementation calculates the class-wise weights after every epoch using the model's class-wise validation accuracy.

4.3 Results on Unbalanced MNIST Binary Classification

Similar to [29], we increase the majority class ratio defined in Equation 8 from 0.9 to 0.995 by increasing the number of training samples of majority class, while keeping the total number of training samples constant at 5000. Following implementation details of [29], we retrain LeNet-5 [31] for each majority class ratio using different loss functions and compare the error rate of the trained model on the test set. Table 1 compares the effect of increasing majority class ratio on the test error rates of LeNet-5, trained using various weighted and unweighted loss functions. For comparison, we use the mean and standard deviation of the classification error rates achieved over 10 runs using random splits. The compared loss functions include (1) Unweighted Cross-Entropy(CE) , which uses an unweighted softmax cross-entropy loss function to train the model; (2) inverse class-frequency weighting (IFW) [23], which uses a weighted softmax cross-entropy loss function where the weight for each class is calculated using the inverse of it's frequency; (3) Focal loss [14], ClassBalanced(CB) loss [15], Equalization loss (EQL) [16] and L2RW [29] are state-of-the-art loss functions.

As can be seen from Table 1, our class-wise difficulty-balanced cross-entropy (CDB-CE) loss function performs better than the others. But to ensure a good performance, it is important to select an appropriate value of τ. Hence we conduct another experiment to investigate the effect of changing τ on the performance of our method. For that, we compare the performance of our CDB-CE loss function for different values of τ and different majority class ratios. The results of the experiment are listed in Table 2.

As can be seen from Table 2, increasing value of τ initially helps in improving the performance of our method but after a certain point, it leads to a drop in the performance. We believe that the drop comes due to the excessive down-weighting of the samples of the easy classes. In Table 2, our method works well over a wide range of τ values. The best value for τ varies even with the majority class ratio. But one interesting thing to notice is that even though dynamically updating τ does not always give the best performance, it's performance is never too far from the best and it consistently outperforms all the existing methods listed in Table 1. Therefore, dynamically updating τ can be a default choice to

Table 1. Mean and standard deviation of classification error rates (%) of LeNet-5 [31] trained for MNIST [1] imbalanced binary classification using different loss functions for different majority class ratios. Here we show the best results obtained by each of the loss functions in our implementation. For class-wise difficulty-balanced softmax cross-entropy (CDB-CE) loss (Ours), we report the results with dynamically updated τ.

Maj. class ratio	0.9	0.95	0.98	0.99	0.995
Unweighted CE	1.50 ± 0.51	2.36 ± 0.46	5.09 ± 0.41	8.59 ± 0.41	14.35 ± 1.10
IFW [23]	1.16 ± 0.40	1.74 ± 0.31	3.13 ± 0.74	6.01 ± 0.56	8.94 ± 0.70
Focal Loss [14]	1.74 ± 0.26	2.78 ± 0.29	6.67 ± 0.63	11.11 ± 1.20	17.17 ± 0.86
CB Loss [15]	1.07 ± 0.23	1.79 ± 0.39	3.58 ± 0.71	5.88 ± 1.20	8.61 ± 1.11
EQL [16]	1.49 ± 0.34	2.26 ± 0.41	2.43 ± 0.14	2.60 ± 0.33	3.71 ± 0.41
L2RW [29]	1.24 ± 0.69	1.76 ± 1.12	2.06 ± 0.85	2.63 ± 0.65	3.94 ± 1.23
CDB-CE(Ours)	$\mathbf{0.74 \pm 0.14}$	$\mathbf{1.27 \pm 0.33}$	$\mathbf{1.65 \pm 0.26}$	$\mathbf{2.39 \pm 0.41}$	$\mathbf{3.71 \pm 0.27}$

Table 2. Mean and standard deviation of classification error rates (%) of LeNet-5 [31] trained for MNIST imbalanced binary classification using our CDB-CE loss with different values of τ For dynamically updating τ (last row), we update the value of τ after every epoch as given in Equation 3.

Maj. class ratio	0.9	0.95	0.98	0.99	0.995
$\tau = 0.5$	1.06 ± 0.34	1.43 ± 0.24	1.93 ± 0.27	2.59 ± 0.44	4.03 ± 0.41
$\tau = 1.0$	0.90 ± 0.27	1.38 ± 0.20	1.86 ± 0.30	2.49 ± 0.57	3.94 ± 0.36
$\tau = 1.5$	0.85 ± 0.19	1.35 ± 0.31	1.71 ± 0.28	2.31 ± 0.38	$\mathbf{3.54 \pm 0.25}$
$\tau = 2.0$	0.75 ± 0.15	1.21 ± 0.32	1.75 ± 0.36	$\mathbf{2.23 \pm 0.34}$	3.65 ± 0.41
$\tau = 5.0$	0.88 ± 0.25	$\mathbf{1.19 \pm 0.36}$	2.00 ± 0.32	2.51 ± 0.41	3.78 ± 0.43
$\tau = 7.0$	0.96 ± 0.20	1.20 ± 0.20	2.04 ± 0.30	2.64 ± 0.40	4.13 ± 0.37
dyn. updated τ	$\mathbf{0.74 \pm 0.14}$	1.27 ± 0.33	$\mathbf{1.65 \pm 0.26}$	2.39 ± 0.41	3.71 ± 0.27

select the value of τ in case one wants to avoid trial and error searching for the best τ.

4.4 Results on Long-Tailed CIFAR-100

We conduct extensive experiments on long-tailed CIFAR-100 dataset [2, 15] as well. ResNet-32 [32] is retrained for different imbalance factors in the training dataset, using different loss functions. Table 3 reports the classification accuracy(%) of each such trained model on the CIFAR-100 test set. We compare the results of our method with that of Focal loss [14], Class-Balanced loss [15], L2RW [29], Meta-Weight Net [36] and Equalization loss [16].

As can be seen from Table 3, our CDB-CE loss with dynamically updated τ provides better performance than the others in most cases. But as stated in

Table 3. Top-1 classification accuracy (%) of ResNet-32 trained on long-tailed CIFAR-100 training data. † means that the result has been copied from the origin paper [15, 36, 37]. For CDB-CE loss (Ours), we report the results with dynamically updated τ.

Imbalance	200	100	50	20	10
Focal loss † [14]	35.62	38.41	44.32	51.95	55.78
Class-Balanced † [15]	36.23	39.60	45.32	52.99	57.99
L2RW † [29]	33.38	40.23	44.44	51.64	53.73
Meta-Weight Net † [36]	**37.91**	42.09	46.74	**54.37**	58.46
Equalization loss[1] † [16]	37.34	40.54	44.70	54.12	58.32
LDAM-DRW † [37]	–	42.04	–	–	58.71
CDB-CE (Ours)	37.40	**42.57**	**46.78**	54.22	**58.74**

Table 4. Top-1 classification accuracy (%) of ResNet-32 trained using class-wise difficulty-balanced cross-entropy (ours) loss function for different values of τ. $\tau = 0$ means the original unweighted softmax cross-entropy loss function.

Imbalance	200	100	50	20	10
$\tau = 0$	34.95	38.21	43.89	51.34	55.65
$\tau = 0.5$	37.21	41.26	46.13	**54.60**	58.29
$\tau = 1.0$	**37.99**	41.67	46.45	53.48	**59.47**
$\tau = 1.5$	37.63	**42.70**	**47.09**	52.74	58.68
$\tau = 2.0$	37.03	42.44	46.94	53.00	58.65
$\tau = 5.0$	36.81	40.62	45.45	51.67	54.48
dynamically updated τ	37.40	42.57	46.78	54.22	58.74

Section 4.3, the results of our method depend highly on the value of τ. Hence, we conduct a further study of how the performance of our method varies on the long-tailed CIFAR-100 dataset with the change in τ. The results are shown in Table 4.

Using $\tau = 0$ makes our CDB-CE loss drop back to the original unweighted softmax cross-entropy loss function. From Table 4, almost a wide range of values for τ helps our weighted loss to get better results than the baseline of $\tau = 0$. Again the interesting thing is even though dynamically updating τ does not give the best results, it's performance is not far from the best and it outperforms existing methods of Table 3 in majority of the cases.

[1] The equalization loss results reported in [16] use more augmentation techniques (e.g., Cutout [38], autoAugment [39]) compared to [15, 36]. Hence for fair comparison, we report the results that we achieved without using the additional augmentation.

Table 5. Top-1 classification accuracy (%) of ResNet-10 on ImageNet-LT for different methods. † means that the result has been copied from the origin paper [16, 20, 28].

Method	Top-1 Accuracy(%)
Focal loss † [14]	30.50
OLTR † [20]	35.60
Joint training † [28]	34.80
Equalization loss † [16]	36.44
OLTR [20] + CDB-CE(Ours)	36.70
Joint training [28]+ CDB-CE(Ours)	37.10
CDB-CE (Ours)	**38.49**

4.5 Results on ImageNet-LT

We compare the performance of our method on ImageNet-LT with other state-of-the-art methods. For comparison, we use the top-1 classification accuracy as our evaluation metric. The results are listed in the Table 5.

For OLTR [20]+CDB-CE and Joint training [28]+CDB-CE, we implemented our method in the original implementations of [20, 28] available on github. From Table 5, it can be seen that our CDB-CE loss not only achieves the best result but it also helps to boost the performance of OLTR and Joint training.

4.6 Results on EGTEA

We also conduct extensive experiments on EGTEA [30] dataset. As shown in Fig. 1, EGTEA dataset is inherently class-imbalanced. The average amount of training samples per class is 1216.0 for the five most frequent classes while for the rest of the 14 classes, the average is only 158.5. That is why we define the five most frequent classes (i.e., 'Take' , 'Put' , 'Open' , 'Cut' and 'Read') together as our 'majority classes' and the rest of the classes as our 'minority classes'. Table 6 reports the results on the test split of a 3D-ResNeXt101, trained on EGTEA dataset using various loss functions. For comparison, we use four different metrics (1) 'Acc@Top1' is the micro-average of the top-1 accuracies of all the classes (2) 'Acc@Top5' is the micro-average of the top-5 accuracies of all the classes (3) 'Recall' is the macro-average of the recall values of all the classes (4) 'Precision' is the macro-average of the precision values of all the classes.

From Table 6, our proposed method achieves significant performance gains in all the metrics compared to other loss functions. In order to ensure that these gains are not entirely because of the majority classes, we conduct a further study, where we compare the performance of different loss functions on the 'majority classes' and 'minority classes' separately. The results are tabulated in Table 7.

Table 7 confirms that our proposed method helps to improve the macro-averaged recall and precision on the 'minority classes'. Though we see a drop in average recall and precision for the 'majority classes' using our method, Table 6

Table 6. 3D-ResNeXt101 results on EGTEA test set. For class-wise difficulty-balanced cross entropy (ours), we use dynamically updated τ.

	Acc@Top1	Acc@Top5	Recall	Precision
Unweighted CE	67.41	95.40	64.77	61.73
Focal loss [14]	64.34	94.36	59.17	59.09
Class-Balanced loss [15]	66.86	95.69	63.26	63.39
CDB-CE (Ours)	**69.14**	**96.84**	**66.24**	**63.86**

Table 7. 3D-ResNeXt101 results on the 'majority classes' and 'minority classes' for different training loss functions. As explained before, the five most frequent classes together constitute the 'majority classes' while the rest of them are the 'minority classes'. We use 'Recall' and 'Precision' for comparison.

	Majority classes		Minority classes	
	Recall	Precision	Recall	Precision
Unweighted CE	74.91	**75.62**	61.14	56.75
Focal loss [14]	70.27	75.00	55.21	53.40
Class-Balanced loss [15]	**75.95**	73.75	58.72	59.68
CDB-CE (Ours)	74.42	73.51	**63.31**	**60.42**

shows that we achieve an overall performance gain for both precision and recall. Therefore, improvement on the 'minority classes' accounts for the overall gain.

5 Conclusion

In this paper, we have proposed a new weighted-loss method for solving class-imbalance. The key idea of our method is to take the difficulty of each class into consideration, rather than the number of training samples of the class, for assigning weights to samples. Based on this idea, we define a quantification for the dynamic difficulty of each class. Further we propose a difficulty-based weighting system that dynamically assigns weights to the samples based on the difficulty of their classes. We also conduct extensive experiments on artificially induced class-imbalanced MNIST, CIFAR and ImageNet datasets and inherently class-imbalanced EGTEA dataset. The experimental results show that using our weighting strategy with cross-entropy loss function helps to boost its performance and achieve best results on imbalanced datasets. Moreover, achieving good results on both image and video datasets show that the benefit of our method is not limited to any particular type of data.

References

1. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine **29** (2012) 141–142
2. Krizhevsky, A.: Learning multiple layers of features from tiny images. University of Toronto (2012)
3. Deng, J., Dong, W., Socher, R., Li, L., Kai Li, Li Fei-Fei: ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. (2009) 248–255
4. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
5. Goyal, R., Ebrahimi Kahou, S., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fruend, I., Yianilos, P., Mueller-Freitag, M., Hoppe, F., Thurau, C., Bax, I., Memisevic, R.: The "something something" video database for learning and evaluating visual common sense. In: The IEEE International Conference on Computer Vision (ICCV). (2017)
6. Soomro, K., Zamir, A.R., Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild. CoRR **abs/1212.0402** (2012)
7. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research **16** (2002) 321–357
8. Bennin, K.E., Keung, J., Phannachitta, P., Monden, A., Mensah, S.: Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. IEEE Transactions on Software Engineering **44** (2018) 534–550
9. Liu, X., Wu, J., Zhou, Z.: Exploratory undersampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **39** (2009) 539–550
10. Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
11. Sohn, K.: Improved deep metric learning with multi-class n-pair loss objective. In Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R., eds.: Advances in Neural Information Processing Systems 29. Curran Associates, Inc. (2016) 1857–1865
12. Huang, C., Li, Y., Loy, C.C., Tang, X.: Learning deep representation for imbalanced classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
13. Wang, Y.X., Ramanan, D., Hebert, M.: Learning to model the tail. In Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., eds.: Advances in Neural Information Processing Systems 30. Curran Associates, Inc. (2017) 7029–7039
14. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. In: The IEEE International Conference on Computer Vision (ICCV). (2017)
15. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)

16. Tan, J., Wang, C., Li, B., Li, Q., Ouyang, W., Yin, C., Yan, J.: Equalization loss for long-tailed object recognition. In: The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (2020)
17. Li, B., Liu, Y., Wang, X.: Gradient harmonized single-stage detector. CoRR **abs/1811.05181** (2018)
18. Han, H., Wang, W.Y., Mao, B.H.: Borderline-smote: A new over-sampling method in imbalanced data sets learning. In Huang, D.S., Zhang, X.P., Huang, G.B., eds.: Advances in Intelligent Computing, Berlin, Heidelberg, Springer Berlin Heidelberg (2005) 878–887
19. Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C.: Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.B., eds.: Advances in Knowledge Discovery and Data Mining, Berlin, Heidelberg, Springer Berlin Heidelberg (2009) 475–482
20. Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B., Yu, S.X.: Large-scale long-tailed recognition in an open world. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
21. Wang, Y.X., Ramanan, D., Hebert, M.: Learning to model the tail. In Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., eds.: Advances in Neural Information Processing Systems 30. Curran Associates, Inc. (2017) 7029–7039
22. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. CoRR **abs/1310.4546** (2013)
23. Huang, C., Li, Y., Loy, C.C., Tang, X.: Learning deep representation for imbalanced classification. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2016) 5375–5384
24. Amin, A., Anwar, S., Adnan, A., Nawaz, M., Howard, N., Qadir, J., Hawalah, A., Hussain, A.: Comparing oversampling techniques to handle the class imbalance problem: A customer churn prediction case study. IEEE Access **4** (2016) 7940–7957
25. Tsai, C.F., Lin, W.C., Hu, Y.H., Yao, G.T.: Under-sampling class imbalanced datasets by combining clustering analysis and instance selection. Information Sciences **477** (2018)
26. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). Volume 2. (2006) 1735–1742
27. Ge, W.: Deep metric learning with hierarchical triplet loss. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018)
28. Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., Kalantidis, Y.: Decoupling representation and classifier for long-tailed recognition. In: Eighth International Conference on Learning Representations (ICLR). (2020)
29. Ren, M., Zeng, W., Yang, B., Urtasun, R.: Learning to reweight examples for robust deep learning. CoRR **abs/1803.09050** (2018)
30. Li, Y., Liu, M., Rehg, J.M.: In the eye of beholder: Joint learning of gaze and actions in first person video. In: The European Conference on Computer Vision (ECCV). (2018)
31. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86** (1998) 2278–2324
32. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015)

33. Xie, S., Girshick, R.B., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. CoRR **abs/1611.05431** (2016)
34. Hara, K., Kataoka, H., Satoh, Y.: Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018)
35. Ketkar, N. In: Introduction to PyTorch. Apress, Berkeley, CA (2017) 195–208
36. Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., Meng, D.: Meta-weight-net: Learning an explicit mapping for sample weighting. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., eds.: Advances in Neural Information Processing Systems 32. Curran Associates, Inc. (2019) 1919–1930
37. Cao, K., Wei, C., Gaidon, A., Arechiga, N., Ma, T.: Learning imbalanced datasets with label-distribution-aware margin loss. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., eds.: Advances in Neural Information Processing Systems 32. Curran Associates, Inc. (2019) 1567–1578
38. Devries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. CoRR **abs/1708.04552** (2017)
39. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation strategies from data. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)