ASYNCHRONOUS MODELING: A DUAL-PHASE PERSPECTIVE FOR LONG-TAILED RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

This work explores deep learning based classification model on real-world datasets with a long-tailed distribution. Most of previous works deal with the long-tailed classification problem by re-balancing the overall distribution within the whole dataset or directly transferring knowledge from data-rich classes to data-poor ones. In this work, we consider the gradient distortion in long-tailed classification when the gradient on data-rich classes and data-poor ones are incorporated simultaneously, i.e., shifted gradient direction towards data-rich classes as well as the enlarged variance by the gradient fluctuation on data-poor classes. Motivated by such phenomenon, we propose to disentangle the distinctive effects of data-rich and data-poor gradient and asynchronously train a model via a dual-phase learning process. The first phase only concerns the data-rich classes. In the second phase, besides the standard classification upon data-poor classes, we propose an exemplar memory bank to reserve representative examples and a memory-retentive loss via graph matching to retain the relation between two phases. The extensive experimental results on four commonly used long-tailed benchmarks including CIFAR100-LT, Places-LT, ImageNet-LT and iNaturalist 2018 highlight the excellent performance of our proposed method.

1 INTRODUCTION

Past years have witnessed huge progress in visual recognition with the successful application of deep convolutional neural networks (CNNs) on large-scale datasets, e.g., ImageNet ILSVRC 2012 (Russakovsky et al., 2015), Places (Zhou et al., 2017). Such datasets are usually artificially collected and exhibit approximately uniform distribution concerning the number of samples in each class. Real-world datasets, however, are always long-tailed that only a few classes occupy the majority of instances in the dataset (data-rich) and most classes have rarely few samples (data-poor) (Reed, 2001; Van Horn & Perona, 2017). When modeling such datasets, many standard methods suffer from severe degradation of overall performance. More specifically, the recognition ability on classes with rarely few instances are significantly impaired (Liu et al., 2019).

One prominent direction is to apply class re-sampling or loss re-weighting to balance the influence of different classes (Byrd & Lipton, 2019; Shu et al., 2019) and another alternative is to conduct transferring (Wang et al., 2017; Liu et al., 2019) by the assumption that knowledge obtained on the data-rich classes should benefit the recognition of data-poor classes. Recently, more sophisticated models are designed to train the model either base on some new findings (Zhou et al., 2020; Kang et al., 2020) or combine all available techniques (Zhu & Yang, 2020). However, the property of long-tailed setting makes it remain to be difficult to achieve large gains compared to balanced datasets.

In contrast to the aforementioned strategies, we approach the long-tailed recognition problem by analyzing gradient distortion in long-tailed data, attributing to the interaction between gradients generated by data-rich and data-poor classes, i.e., the direction of overall gradient is shifted to be closer to the gradient on data-rich classes and its norm variance is increased due to the dramatic variation in the gradient generated by data-poor classes. The degenerated performance when comparing with balanced datasets indicates the gradient distortion is negative during model training. Motivated by this, we hypothesize that the combined analysis for gradients generated by data-rich and data-poor classes could be improper in long-tailed data and attempt to disentangle these two gradients. We thus propose the conception of asynchronous modeling and split the original network to promote a dual-phase learning, along with the partition of the given dataset. In phase I, data-rich classes keeps the bulk of the original dataset. It facilitates better local representation learning and more precise classifier boundary determination by eliminating the negative gradient interaction produced by data-poor classes. Based on the model learned in phase I, we involve the rest data to do new boundary exploration in the second phase.

While transiting from the first phase to the second, it is hoped to reserve the knowledge learned in the first phase. Specifically, we design an exemplar memory bank and introduce a memory-retentive loss. The memory bank reserves a few most prominent examples from classes in the first phase and collaborates with data in the second phase for classification. Also, the collaborated data, together with the new memory-retentive loss, tries to preserve old knowledge when the model adapts to new classes in the second phase.

In the experiments, we evaluate the proposed asynchronous modeling strategy by comparing to typical strategies, which include the re-balancing based methods (Cao et al., 2019) and transferring based methods (Liu et al., 2019). Furthermore, we also consider the latest, more sophisticated works, like BBN (Zhou et al., 2020), IEM (Zhu & Yang, 2020). The comprehensive study and comparison across four commonly used long-tailed benchmarks, including CIFAR100-LT, Places-LT, ImageNet-LT and iNaturalist 2018 validate the efficacy of our method.

2 Related Work

Class re-sampling. Most works along with this line can be categorized as over-sampling of tail classes (Chawla et al., 2002; Han et al., 2005; Byrd & Lipton, 2019) or under-sampling over head classes (Drummond et al., 2003). While the idea of re-sampling makes the overall distribution more balanced, it may encounter the problem of over-fitting on rare data and the missing of critical information on dominant classes (Chawla et al., 2002; Cui et al., 2019), thus hurting the overall generalization. Beyond that, Ouyang et al. (2016); Liu et al. (2019) also involve a more refined idea of fine-tuning after representation extraction to adjust the final decision boundary.

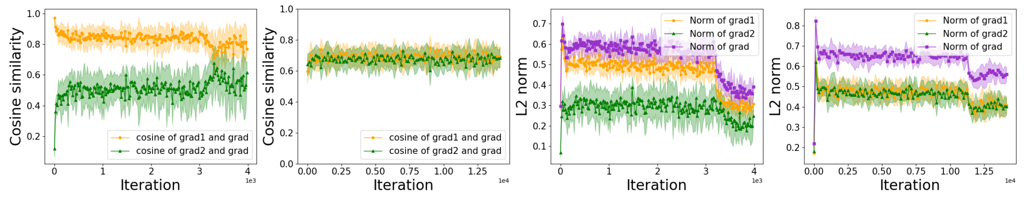
Loss re-weighting. Methods based on loss re-weighting generally allocate larger weights for tail classes to increase their importance (Lin et al., 2017; Ren et al., 2018; Shu et al., 2019; Cui et al., 2019; Khan et al., 2017; 2019; Huang et al., 2019). However, direct re-weighting method is difficult to be optimized when tackling a large-scale dataset (Mikolov et al., 2013). Recently, Cao et al. (2019) considers the margins of the training set and introduces a label-distribution-aware loss to enlarge the margins of tail classes. Hayat et al. (2019) proposes the first hybrid loss function to jointly cluster and classify feature vectors in the Euclidean space and to ensure uniformly spaced and equidistant class prototypes.

Knowledge transfer. Along this line, methods based on knowledge transfer handle the challenge of imbalanced dataset by transferring the information learned on head classes to assist tail classes. While Wang et al. (2017) proposes to transfer meta-knowledge from the head in a progressive manner, recent strategies take consideration of intra-class variance (Yin et al., 2019), semantic feature (Liu et al., 2019; Chu et al., 2020) or domain adaptation (Jamal et al., 2020).

Recently, BBN (Zhou et al., 2020) and LWS (Kang et al., 2020) boost the landscape of long-tailed problem based on some insightful findings. The former asserts that prominent class re-balancing methods can impair the representation learning and the latter claims that data imbalance might not be an issue in learning high-quality representations. IEM (Zhu & Yang, 2020) designs a more complex model that tries to concern available techniques, like feature transferring and attention. In this paper, we are motivated by gradient distortion in long-tailed data, which is caused by the gradient interaction between data-rich classes and data-poor classes. We thus propose to split the learning stage into two phases. We demonstrate that this separation allows straightforward approaches to achieve high recognition performance, without introducing extra parameters.

3 Our Method

Let $X = \{\boldsymbol{x}_i, y_i\}, i \in \{1, ..., n\}$ be the training set, where $\boldsymbol{x}_i$ is the training data and y_i is its corresponding label. The number of instances in class j is denoted as n_j and the total number of



(a) CIFAR100-LT (b) vanilla CIFAR100 (c) CIFAR100-LT (d) vanilla CIFAR100

Figure 1: 'grad1'('grad2'): gradient generated by data-rich (data-poor) classes in CIFAR100-LT or gradient generated by the same classes in vanilla CIFAR-100; 'grad': the overall gradient in both datasets. **(a)** and **(b)**: Cosine similarity between grad1 and grad, grad2 and grad; **(c)** and **(d)**: Norm of grad1, grad2, and grad; **(a)** and **(b)** indicate the overall gradient is shifted to be closer to the direction of gradient generated by data-rich class. **(c)** and **(d)** show that the variance of overall gradient is enlarged by the fluctuation of gradient on data-poor classes.

training samples is denoted as $n = \sum_{j=1}^{C} n_j$, where C is the number of classes. Without loss of generality, we assume that the classes are sorted in decreasing order, that is, if $i > j$, $n_i \leq n_j$. We define the whole network as $f(\boldsymbol{x}; [\boldsymbol{W}_r; \boldsymbol{W}_c])$, where f is the implemented deep learning model with parameters $\boldsymbol{W}_r$ for representation learning and parameters $\boldsymbol{W}_c$ for classification, and $\boldsymbol{x}$ is the input.

3.1 Gradient Distortion in Long Tail

Given a long-tailed dataset, our goal is to achieve better overall performance across all classes. In contrast to previous common heuristics (e.g., resampling, reweighting and feature transfer), we revisit the problem of long-tailed classification from the perspective of gradient distortion. The overall gradient for updating is modulated by the gradients generated by data-rich classes in the head and data-poor classes in the tail. To state the details, we visualize the associated metrics in the training process of vanilla CIFAR100 and long-tailed CIFAR100 (CIFAR100-LT) in Fig. 1. Specifically, the cosine similarity between the gradients is visualized in Fig. 1(a) (CIFAR100-LT) and Fig. 1(b) (vanilla CIFAR100). Similarly, the norm of each gradient is recorded in Fig. 1(c) (CIFAR100-LT) and Fig. 1(d) (vanilla CIFAR100). The higher similarity between the overall gradient and the data-rich gradient indicates that the overall gradient is shifted to the direction of the data-rich gradient. Meanwhile, the norm variance of overall gradient is enlarged due to more dramatic fluctuation of the gradient on data-poor classes. Motivated by the degenerated performance in long-tailed dataset, it is hypothesized that synchronous application of two distinctive gradients could impair the overall performance.

3.2 Asynchronous Modeling

Rather than directly regulating the overall gradient as previous methods, we begin with the disentanglement of two gradients and propose a dual-phase asynchronous modeling strategy. The data from data-rich classes is first considered in model training and then the rest classes are involved. Such asynchronous operation not only reduces the potential disturbance between two gradients, but also ensures the benefits of each gradient to be exploited. Mathematically, the original dataset is X with C classes. Suppose C_1 classes are considered in phase I, we then write X_1 as the set of data from C_1 classes. The data in rest C_2 classes is denoted as X_2, where $C_2 = C - C_1$. Accordingly, the parameters $\boldsymbol{W}_c$ for C classes in $f(\boldsymbol{x}; [\boldsymbol{W}_r, \boldsymbol{W}_c])$ are truncated as $\boldsymbol{W}_c^1$ for C_1 classes in the first phase.

3.2.1 Learning in the First Phase

In model learning from data X_1, the consideration of gradient on data-poor categories is avoided, which keeps the truncated model $f(\boldsymbol{x}; [\boldsymbol{W}_r; \boldsymbol{W}_c^1])$ to be more concentrated. In optimization, the cross-entropy loss over the classes in X_1 is minimized with respect to parameters $\boldsymbol{W}_r$ and $\boldsymbol{W}_c^1$.

$$\mathcal{L}_1 = -\sum_{(\boldsymbol{x},y)\in X_1} y \log f(\boldsymbol{x}; [\boldsymbol{W}_r; \boldsymbol{W}_c^1]). \tag{1}$$

For further improvement in the training, some balanced sampling strategies could be incorporated in this phase. For example, the progressively-balanced strategy in (Kang et al., 2020) combines instance-balanced sampling and class-balanced sampling, that is, $p_j(t) = (1 - \frac{t}{T})\frac{n_j}{\sum_{i=1}^{C} n_i} + \frac{t}{T}\frac{1}{C}$, where $p_j(t)$ denotes the sampling probability for class j in training epoch t. It is computed with linear combination for instances-based probability $\frac{n_j}{\sum_{i=1}^{C} n_i}$ and class-based probability $\frac{1}{C}$. T is the total epoch number.

3.2.2 Joint prediction in the second phase

We wish to involve the data in X_2 to obtain a complete model across all C classes for overall evaluation. To do so, on the basis of parameters $\boldsymbol{W}_r$ obtained in phase I, we introduce the classifier parameters $\boldsymbol{W}_c^2$ for the recognition of new classes in X_2. Similar to phase I, the standard cross-entropy loss across all data in X_2 is considered. However, considering solely on data X_2 tends to forget the knowledge learned in the first phase. To tackle with the obstacle, we thus design a memory bank and memory-retentive loss to realize the seamless connection between two data splits. First, representative samples in X_1 are retained in an augmented memory module to enable the joint prediction over all classes. Second, the examples reserved in the memory are combined with X_2, which are collaboratively trained with a unified memory-retentive loss.

Exemplar memory bank. In maintaining the knowledge obtained in the first phase, we design an exemplar memory bank that selects only a few most representative samples from classes in X_1. For simplicity, the number of selected samples from each class is set to be equal. We denote the reserved data in the memory bank as M. Ideally, the most representative examples are samples that are closest to the center of each class. However, a precise class center is not always accessible. Thus in practice, the center is progressively estimated by accessing over the entries generated in previous steps to infer new entry in the memory bank.

Without loss of generalization, we consider class j in dataset X_1 to demonstrate the detailed operation. We first compute the average feature from all examples in class j in original training set X_1 to serve as a class prototype c_j, which is thus the initial estimation of class center. We return the instance which is closest to c_j in X_1 and set it as the first selected sample for the memory bank,

$$m_1 = \arg\max_{x_i \in X_1} \{s(c_j, X_1)\}, \tag{2}$$

where s is a vector space similarity metric, like cosine similarity. m_1 is used to denote the returned sample x_i. Before selecting the rest instances from X_1, we need to update the estimated center c_j. Without loss of generality, suppose we have selected k samples from X_1 and denote the feature map of data in memory bank as $M_j = [m_1, m_2, .., m_k] \in \mathbb{R}^{k \times d}$, where d is the dimension of each feature map. Each sample in M_j serves as a guided hypothesis and its correlation with c_j can then be computed for the new state z_{k+1}, that is,

$$p_i = \frac{\exp(s(c_j, m_i))}{\sum_i \exp(s(c_j, m_i))}, \tag{3}$$

$$z_{k+1} = \sum_{i=1}^{k} p^i m_i = pM_j, \tag{4}$$

where s is the same similarity metric as above. p_i is computed by the distances between the selected data and the center prototype and it serves as weights to update state z_{k+1}. z_{k+1} is the weighted average of all feature maps in M_j. New samples can then be returned for $k+1$ step by performing

$$m_{k+1} = \arg\max_{x_i \in X_1} \{s(c_j + \Delta, X_1)\}, \tag{5}$$

where Δ is the residual between c_j and z_{k+1}, i.e., $\Delta = c_j - z_{k+1}$. m_{k+1} is used to denote the returned sample x_i.

Memory-retentive loss. Based on the memory bank, we obtain a combined data set D by extending X_2 with examples in the memory bank M, i.e., $D = M \bigcup X_2$. Similarly, the joint prediction with a cross-entropy loss is first considered. When the model is adapted to fit data X_2, the knowledge learned in X_1 tends to be forgotten. We thus introduce a new memory-retentive loss $L_{dis}^{\mathcal{G}}$ based on graph matching, which provides a strong constraint in memorizing previous knowledge. Specifically, the feature map of each data in the training set D is a node in a graph. Based on the model learned in the first stage and the new model to be trained in the second phase, two graphs $\mathcal{G}_{old}$ and $\mathcal{G}_{new}$ can thus be constructed. That is, we not only consider feature similarity of a single example on the old model and the new model, but also compute the global matching similarity on the whole training set D. Suppose the feature map of one node in $\mathcal{G}_{old}$ is $\boldsymbol{z}_i$ and in $\mathcal{G}_{old}$ is $\hat{\boldsymbol{z}}_i$, thus the similarity between old graph $\mathcal{G}_{old}$ and new graph $\mathcal{G}_{new}$ is measured by computing the change between any node $\boldsymbol{z}_i$ in $\mathcal{G}_{old}$ and any node $\boldsymbol{z}_j$ in $\mathcal{G}_{new}$, that is,

$$a_{ji} = \frac{\exp(s(\boldsymbol{z}_i, \hat{\boldsymbol{z}}_j))}{\sum_{j'} \exp(s(\boldsymbol{z}_i, \hat{\boldsymbol{z}}_{j'}))}, \boldsymbol{z}_i \in \mathcal{G}_{old}, \hat{\boldsymbol{z}}_j, \hat{\boldsymbol{z}}_{j'} \in \mathcal{G}_{new}, \tag{6}$$

$$\mu_i = \sum_j a_{ji} \|\boldsymbol{z}_i - \hat{\boldsymbol{z}}_j\|, \boldsymbol{z}_i \in \mathcal{G}_{old}, \hat{\boldsymbol{z}}_j \in \mathcal{G}_{new}, \tag{7}$$

$$L_{dis}^{\mathcal{G}} = \sum_i \mu_i, \boldsymbol{z}_i \in \mathcal{G}_{old}, \tag{8}$$

where s is the vector similarity metric. a_{ji} represents the distance between node i in graph $\mathcal{G}_{old}$ and node j in graph $\mathcal{G}_{new}$, μ_i thus ntuitively measures the difference between $\boldsymbol{z}_j$ and its closest neighbor in graph $\mathcal{G}_{new}$. Consider all nodes in graph $\mathcal{G}_{old}$ together, we obtain the memory-retentive loss $L_{dis}^{\mathcal{G}}$, which describe graph similarity between two graphs.

Overall loss. Combined the above analysis together, the overall loss in phase II is thus as below:

$$\mathcal{L} = \frac{1}{|D|} \sum_{\boldsymbol{x} \in D} (L_{cls}(\boldsymbol{x}) + L_{intra}(\boldsymbol{x})) + \lambda L_{dis}^{\mathcal{G}}, \tag{9}$$

where the first term is for classification and the second is the designed loss which constrains the knowledge in old model through graph matching, λ is a hyperparameter to balance the two terms. Notice that, apart from the standard cross-entropy loss $L_{cls}(\boldsymbol{x})$ for input $\boldsymbol{x}$ in the first term, we also consider an intra-classification loss $L_{intra}(\boldsymbol{x})$ to avoid memory data in M being dominated by new classes in X_2. When we consider cosine linear classifier, one of the instantiations could be $L_{intra}(\boldsymbol{x}) = \sum_{k=1}^{K} \max(0, m - \langle \bar{\boldsymbol{w}}, \bar{\boldsymbol{z}}(\boldsymbol{x}) \rangle + \langle \bar{\boldsymbol{w}}^k, \bar{\boldsymbol{z}}(\boldsymbol{x}) \rangle$, in which, $\bar{\boldsymbol{w}}$ is the ground-truth class embedding and $\bar{\boldsymbol{w}}^k$ denotes the other class embedding, $\bar{\boldsymbol{z}}(\boldsymbol{x})$ is the normalized feature map of $\boldsymbol{x}$, m is a margin value. $\langle \bar{\boldsymbol{w}}, \bar{\boldsymbol{z}}(\boldsymbol{x}) \rangle$ denotes a positive score between $\bar{\boldsymbol{w}}$ and $\bar{\boldsymbol{z}}(\boldsymbol{x})$, while $\langle \bar{\boldsymbol{w}}^k, \bar{\boldsymbol{z}}(\boldsymbol{x}) \rangle$ denotes the negative score between $\bar{\boldsymbol{w}}^k$ and $\bar{\boldsymbol{z}}(\boldsymbol{x})$. L_{intra} optimizes the network to maintain a margin of m between the positive score and the highest negative score. Finally, for a comprehensive overview of the asynchronous modeling structure, we can find it in Algorithm 1 in Appendix B.

4 Experiments

Datasets. We perform extensive experiments on four long-tailed datasets, including CIFAR100-LT (Cao et al., 2019), Places-LT (Liu et al., 2019), ImageNet-LT (Liu et al., 2019), and iNaturalist 2018 (iNatrualist, 2018). CIFAR100-LT is created with three different imbalance factors 50, 100, 200. For different versions of CIFAR100-LT, they are created from the original CIFAR100 that the samples in class y are truncated to $n_y \mu^{\frac{y}{c-1}}$, where c is the number of all classes, y is the index of class and n_y is the original number of training examples in class y. By varying μ to be 0.02, 0.01, 0.005, we obtain three groups of CIFAR100-LT with imbalance factor 50, 100, 200. More dataset details can be found in Appendix A.

Evaluation Metrics. We evaluate the models on the corresponding balanced test/validation datasets and report the overall top-1 accuracy over all classes, denoted as *Overall*. Furthermore, to better describe the internal diversity across classes with different training samples, we follow Liu et al. (2019) to split the given dataset into three disjoint sets: *Many-shot* (classes with more than 100 images), *Medium-shot* (20$\sim$100 images) and *Few-shot* (fewer than 20 images) and report the corresponding accuracy for comparison.

4.1 Comparison with state-of-the-art

In this section, we compare our method with a wild range of previous works in addressing long-tailed classification from different directions.

Table 1: Evaluation results on Places-LT. † denotes our result with extended parameters.

Method	**Many**	**Medium**	**Few**	**Overall**
Joint (Kang et al., 2020)	45.7	27.3	8.2	30.2
Lifted Loss (Oh Song et al., 2016)	41.1	35.4	24.0	35.2
Focal loss (Lin et al., 2017)	41.1	34.8	22.4	34.6
Range Loss (Zhang et al., 2017b)	41.1	35.4	23.2	35.1
FSLwF (Gidaris & Komodakis, 2018)	43.9	29.9	29.5	34.9
OLTR (Liu et al., 2019)	44.7	37.0	25.3	35.9
NCM (Kang et al., 2020)	40.4	37.1	27.3	36.4
cRT (Kang et al., 2020)	42.0	37.6	24.9	36.7
LWS (Kang et al., 2020)	40.6	39.1	28.6	37.6
IEM (Zhu & Yang, 2020)	46.8	39.2	28.0	39.7
Ours (plain)	46.2	36.3	24.5	37.6
Ours	44.2	40.2	30.9	39.8
Ours†	44.5	40.9	31.8	40.4

Table 2: Evaluation on ImageNet-LT with different backbones. † denotes our result with extended parameters.

Method	**ResNet-10**	**ResNet-50**	**ResNet-152**
FSLwF (Gidaris & Komodakis, 2018)	28.4	-	-
Focal Loss (Lin et al., 2017)	30.5	-	-
Range Loss (Zhang et al., 2017b)	30.7	-	-
Lifted Loss (Oh Song et al., 2016)	30.8	-	-
FSA (Chu et al., 2020)	35.2	-	-
IEM (Zhu & Yang, 2020)	43.2	-	-
OLTR (Liu et al., 2019)	35.6	36.7	43.2
Joint (Kang et al., 2020)	34.8	41.6	44.9
NCM (Kang et al., 2020)	35.5	44.3	47.8
cRT (Kang et al., 2020)	41.8	47.3	50.1
LWS (Kang et al., 2020)	41.4	47.7	50.5
Ours (plain)	41.3	49.7	51.7
Ours	42.0	51.0	53.0
Ours†	43.8	52.2	53.8

Places-LT. We initialize the ResNet-152 backbone with ImageNet pre-trained parameters following Kang et al. (2020). In Table 1, we report the result of our baseline without asynchronous modeling and denote it as *Ours (plain)*, that is, considering the dataset together without distinguishing the head and tail. The result based on asynchronous modeling is denoted as *Ours*. In order to compare with baselines like Zhu & Yang (2020), in which more parameters are introduced, we also consider the upgraded version *Ours*† with extended parameters. By comparing our asynchronous modeling with the plain baseline, we notice that the introduction of asynchronous modeling improves the overall result notably. We also outperform the state-of-the-art methods, including OLTR (Liu et al., 2019), LWS (Kang et al., 2020), etc. In comparison with IEM (Zhu & Yang, 2020), we see that comparable result is achieved without introducing any extra parameters. With more parameters considered, much higher accuracy is achieved in our setting.

ImageNet-LT. For ImageNet-LT, the most commonly adopted architecture is ResNet-10. We also evaluate with different backbones for a thorough comparison to previous works. Table 2 shows the overall results on three different backbones, i.e., ResNet-10, ResNet-50 and ResNet-152. We find that our asynchronously obtained model achieves the top performance with impressive improvements over decoupled methods cRT, NCM and LWS in Kang et al. (2020) across all backbones. Also, when comparing with OLTR (Liu et al., 2019) which also applies the memory mechanism, the memory bank in our strategy is obviously more efficient and useful. What is more, our method also outperforms IEM (Zhu & Yang, 2020) when more parameters are considered. More detailed results, i.e., the performance on three splits can be found in Appendix C.

CIFAR100-LT. We follow Cao et al. (2019) and consider three different long–tailed versions with imbalance factors 50, 100, 200. The results in Table 3 demonstrate that in comparison with state-of-the-arts including CB-Focal (Cui et al., 2019), LDAM (Cao et al., 2019) and BBN (Zhou et al., 2020), our method consistently achieves the best performance across all three versions. Especially for CIFAR100-LT with imbalance factor 100, the incorporation of asynchronous modeling introduces more than 2% gains over our plain baseline.

Table 3: The overall results of CIFAR100-LT under different balance factors (200, 100, 50).

Imbalance factor	**200**	**100**	**50**
Cross Entropy	35.63	38.32	43.85
Focal loss (Lin et al., 2017)	34.69	38.41	44.32
Mixup (Zhang et al., 2017a)	36.20	39.54	44.99
CB-Focal (Cui et al., 2019)	36.23	39.60	45.32
LDAM (Cao et al., 2019)	38.06	42.04	46.62
BBN (Zhou et al., 2020)	37.93	42.56	47.02
Ours (plain)	39.35	42.48	47.61
Ours	40.53	44.79	49.32

iNaturalist 2018. We further evaluate our methods on iNaturalist 2018. iNaturalist 2018 is a real-world long-tailed dataset, consisting of over 8K categories. We follow Kang et al. (2020) to train the network for 200 epochs and show the results of two backbones, i.e., ResNet-50 and ResNet-152. From Table 4 we see the results are consistent with the previous datasets: training with asynchronous modeling strategy performs best across different backbones. It not only achieves better results than loss re-weighting or transferring based methods (Cao et al., 2019; Chu et al., 2020) but also outperforms decoupled cRT, NCM, LWS (Kang et al., 2020).

Table 4: Evaluation results on iNaturalist 2018.

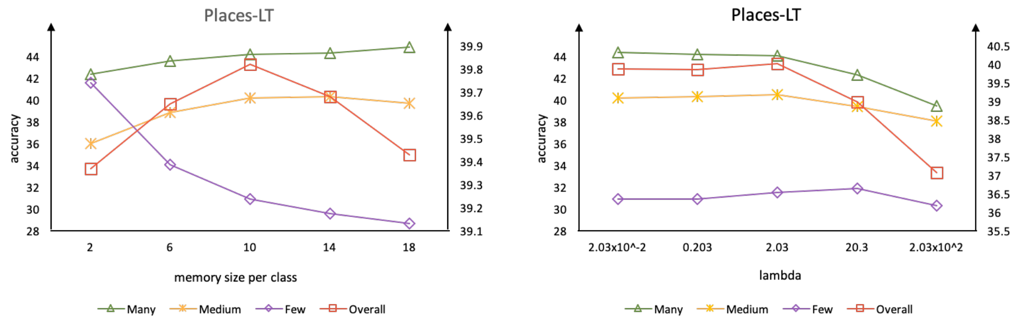
Method	**ResNet-50**	**ResNet-152**
CB-Focal (Cao et al., 2019)	61.1	-
LDAM (Cao et al., 2019)	68.0	-
BBN (Zhou et al., 2020)	69.6	-
NCM (Kang et al., 2020)	63.1	67.3
Joint (Kang et al., 2020)	65.8	69.0
FSA (Chu et al., 2020)	65.9	69.1
cRT (Kang et al., 2020)	67.6	71.2
LWS (Kang et al., 2020)	69.5	72.1
Ours	69.8	72.5

5 Ablation Study

We now perform ablation study to investigate the effect of specific modules. We use ResNet-152 as the backbone and conduct related experiments on Places-LT to study the size of exemplar memory bank and the ratio between the classification loss and the memory-retentive loss. We consider the result under separated $\{$*Many, Medium, Few*$\}$-shots and the overall result. Similarly in Fig. 2 and Fig. 3, the axis for describing different shots is in the left. The change of overall result is depicted in the right of the figure, which is an independent axis.

Size of memory bank. We first explore the effect of memory bank with different sizes. In the experiment, the size of memory bank depends on the selected number of samples from each class. Particularly, we consider five cases and set the reserved number of samples from each class in X_1 as 2, 6, 10, 14, 18, respectively. For each cases, other operations are kept as the same. From Fig. 2(a), we see that with the increment of memory size, the performance on *Few*-shot is decreasing, which is opposite to the result on *Many*-shot. Generally speaking, the best overall result is achieved when memory size equals to 10. We notice that the overall result is changed under different memory sizes, but it is rather stable, varying from 39.4 to 39.8.

The ratio between the classification loss and the memory-retentive loss. Similarly, we also study how the ratio between the classification loss and the memory-retentive loss affects the final results. In practice, such balance is controlled by parameters λ in Eq. 9. Based on the initial option $\lambda = \sqrt{C/C_1}$, in which C is the number of all class and C_1 is the class used in phase I, the initial λ is scaled properly to obtain other four values. As shown in Fig. 2(b), we conclude that the best overall result of Places-LT is achieved when λ equals to 2.03. More importantly, the overall performance retains good for a wild range of λ, i.e., $\lambda \leq 2.03$.



(a) (b)

Figure 2: The classification results on Places-LT with backbone ResNet-152. The right y-axis is to depict the overall result and the left one is for {*Many, Medium, Few*}-shots in each figure. **(a)**: The change of classification results under different memory bank size. **(b)**: The change of classification results under different λ, which balances the classification loss and memory-retentive loss.

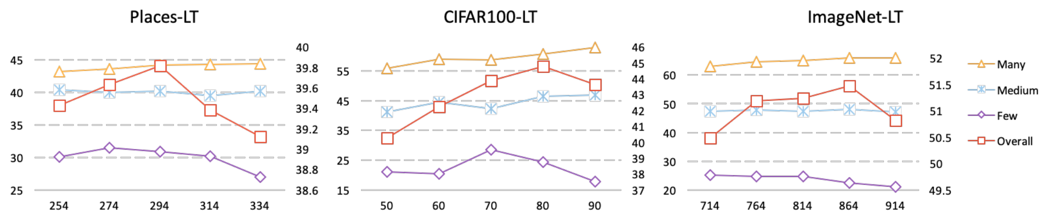


Figure 3: The classification results on three datasets with different disentanglement points. The right y-axis is for the overall performance and the left y-axis is for results on {*Many, Medium, Few*}-shots. With the movement of the disentanglement point, the overall result first increases then decreases.

Through the above analysis of memory bank size and the λ, we notice that the changes of different modules do affect the overall performance. However, the mild variation indicates that our method is robust and stable.

Influence of different partitions. In this part, we investigate the influence of disentanglement point on the final performance. The disentanglement point also corresponds to the index of class since we arrange the order of classes by the number of instance in the paper. We conduct experiments on three datasets, including CIFAR100 with imbalance factor 100, Places-LT and ImageNet-LT and explore five disentanglement points for each dataset. The final results are shown in Fig. 3. To better show the variation of overall performance (the red line), we depict it using a separated vertical axis (the right one in each figure). We also show the change of different shots in each dataset: *Many-shot* in orange, *Medium-shot* in blue and *Few-shot* in purple. From the comparison on three datasets, we conclude the best disentanglement point for each dataset.

6 Conclusion

In this paper, we begin with the visual phenomenon of gradient distortion in long tail and propose an asynchronous modeling strategy that learns a unified recognition model through two phases to better exploit the gradients generated by data-rich classes and data-poor classes. In unifying the training process, we introduce a memory bank and a memory-retentive loss to retain the knowledge learned in the first phase while exploring new boundaries in the second phase. The extensive results on four long tailed benchmark datasets which significantly outperform previous works validate the superior efficacy of our method.

REFERENCES

Jonathon Byrd and Zachary Lipton. What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning*, pp. 872–881, 2019.

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, pp. 1567–1578, 2019.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Peng Chu, Xiao Bian, Shaopeng Liu, and Haibin Ling. Feature space augmentation for long-tailed data. *arXiv preprint arXiv:2008.03673*, 2020.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9268–9277, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pp. 1–8. Citeseer, 2003.

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4367–4375, 2018.

Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pp. 878–887. Springer, 2005.

Munawar Hayat, Salman Khan, Waqas Zamir, Jianbing Shen, and Ling Shao. Max-margin class imbalanced learning with gaussian affinity. *arXiv preprint arXiv:1901.07711*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Chen Huang, Yining Li, Change Loy Chen, and Xiaoou Tang. Deep imbalanced learning for face recognition and attribute prediction. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

iNatrualist. The inaturalist 2018 competition dataset. . *https://github.com/visipedia/inat comp/tree/master/2018*, 2018.

Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7610–7619, 2020.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *Eighth International Conference on Learning Representations (ICLR)*, 2020.

Salman Khan, Munawar Hayat, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Striking the right balance with uncertainty. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 103–112, 2019.

Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573–3587, 2017.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2537–2546, 2019.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4004–4012, 2016.

Wanli Ouyang, Xiaogang Wang, Cong Zhang, and Xiaokang Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 864–873, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.

William J Reed. The pareto, zipf and other power laws. *Economics letters*, 74(1):15–19, 2001.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pp. 1919–1930, 2019.

Grant Van Horn and Pietro Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems*, pp. 7029–7039, 2017.

Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Feature transfer learning for face recognition with under-represented data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5704–5713, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017a.

Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5409–5418, 2017b.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9719–9728, 2020.

Linchao Zhu and Yi Yang. Inflated episodic memory with region self-attention for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4344–4353, 2020.

A Appendix

Dataset Details. Places-LT and ImageNet-LT are artificially truncated to follow a long-tailed distribution from Places-2 (Zhou et al., 2017) and ImageNet-2012 (Deng et al., 2009), respectively. Places-LT contains 62.5K images from 365 categories and the number of images per class varies from 4980 to 5. ImageNet-LT has 115.8 samples from 1000 classes and the number of images per class is decreased from 1280 to 5 images. iNaturalist 2018 is a real-world visual recognition dataset, that naturally exhibits long-tailed distribution. It consists 435,713 samples from 8,142 species.

Implementation Details. We use the platform of PyTorch (Paszke et al., 2019) for all experiments. For CIFAR100-LT, we adopt ResNet-32 as the backbone. The batch size is 64 and the learning rate is initialized with 0.1. The number of epoch for training is 200 and we decay the learning rate at the 160th and 180th epochs by 0.01. For Places-LT, we choose ResNet-152 as the backbone with pre-trained parameters from ImageNet 2012. The learning rate for representation learning is initialized with 5e-4 and that for classifier is 0.05. We train the model for 60 epochs and all the learning rate is decayed at 20th and 40th epochs by 0.01. On ImageNet-LT, we report results with ResNet-10, 50,101,152 (He et al., 2016). Similarly, ResNet-50, 152 are also used for iNaturalist 2018. For ImageNet and iNaturalist 2018, the learning rate is initialized with 0.05 and cosine learning rate scheduler (Loshchilov & Hutter, 2016) is applied to gradually decay learning rate from 0.05 to 0. For all experiments, if not specified, we use SGD optimizer with momentum 0.9, weight decay 5e-4. The image resolution for CIFAR100-LT is 32×32 and the rest is 224×224. The λ is empirically set based on $\sqrt{\frac{num\ old}{num\ new}}$, where "num old" indicates the number of classes in the first stage and "num new" is the number of new classes in the second stage. The threshold to split the dataset is set as the sum of classes in Many- and Medium-shot. For CIFAR100-LT, the threshold is 70, which means that we first learn the 70 classes in the head and then involve the rest. For ImageNet-LT, the threshold is 864 and Places-LT, the threshold is 294.

B Appendix

Algorithm 1 Asynchronous Modeling for Long-Tailed Recognition

```
Input: Dataset $X = \{\boldsymbol{x}_i, y_i\}$, learning rate $\eta$, training epoch $T$;
 1: Divide dataset $X$ into two parts according to the number of instances in each class. The one
    covered data-rich classes is $X_1$ and the rest is $X_2$.
 2: Model parameters $\boldsymbol{W}_1 = [\boldsymbol{W}_r; \boldsymbol{W}_c^1]$ in phase I;
 3: for $i = 1, 2, \cdots, T$ do
 4:     Sample mini-batch $B$ from training set $X_1$;
 5:     Compute cross-entropy loss $\mathcal{L}_1$ on $B$;
 6:     Update overall parameters $\boldsymbol{W}_1 \leftarrow \boldsymbol{W}_1 - \eta \nabla_{\boldsymbol{W}_1} \mathcal{L}_1$;
 7: end for
 8: Construct memory bank with a few samples from classes in $X_1$ and denote the set as $M$;
 9: Update training set $D = X_2 \bigcup M$, extend model parameters as $\boldsymbol{W} = [\boldsymbol{W}_r; \boldsymbol{W}_c]$;
10: for $i = 1, 2, \cdots, T$ do
11:     Sample mini-batch $B$ from training set $D$;
12:     Compute classification loss $\frac{1}{|B|} \sum_{\boldsymbol{x} \in B} (L_{cls}(\boldsymbol{x}) + L_{intra}(\boldsymbol{x}))$;
13:     Compute memory-retentive loss $L_{dis}^{\mathcal{G}}$ on $B$;
14:     Compute overall loss $\mathcal{L} = \frac{1}{|B|} \sum_{\boldsymbol{x} \in B} (L_{cls}(\boldsymbol{x}) + L_{intra}(\boldsymbol{x})) + \lambda L_{dis}^{\mathcal{G}}$;
15:     Update $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \nabla_{\boldsymbol{W}} \mathcal{L}$;
16: end for
Output: Model with parameters $\boldsymbol{W}$.
```

C Appendix

In Table 5, the detailed results of {*Many, Medium, Few*}-shots on ImageNet-LT are described. Besides from ResNet-{50, 152}, ResNet-101 is also considered here. Compared to the baseline without asynchronous learning (Ours (plain)), our method sacrifices little in *Many*-shot but improves a lot in *Medium*- and *Few*-shot. More importantly, we see that our asynchronous strategy boosts the overall performance across all backbones.

Table 5: Comprehensive results on ImageNet-LT with different backbones.

Backbone	**Method**	**Many**	**Medium**	**Few**	**Overall**
ResNet-50	Joint (Kang et al., 2020)	64.0	33.8	5.8	41.6
	NCM (Kang et al., 2020)	53.1	42.3	26.5	44.3
	cRT (Kang et al., 2020)	58.8	44.0	26.1	47.3
	LWS (Kang et al., 2020)	57.1	45.2	29.3	47.7
	Ours (plain)	66.3	45.0	19.2	49.7
	Ours	64.6	48.3	22.1	51.0
ResNet-101	Joint (Kang et al., 2020)	66.6	36.8	7.1	44.2
	NCM (Kang et al., 2020)	56.8	45.1	28.8	47.4
	cRT (Kang et al., 2020)	61.6	46.5	28.0	49.8
	LWS (Kang et al., 2020)	60.1	47.6	31.2	50.2
	Ours (plain)	67.8	46.5	20.6	51.2
	Ours	65.8	49.1	23.5	52.1
ResNet-152	Joint (Kang et al., 2020)	66.9	27.7	7.7	44.9
	NCM (Kang et al., 2020)	56.9	45.6	29.9	47.8
	cRT (Kang et al., 2020)	61.8	46.8	28.4	50.1
	LWS (Kang et al., 2020)	60.6	47.8	31.4	50.5
	Ours (plain)	67.9	47.3	21.5	51.7
	Ours	66.5	50.6	23.6	53.0