Memory-Based Jitter: Improving Visual Recognition on Long-Tailed Data with Diversity in Memory

Jialun Liu[1,2*], **Wenhui Li**[1†], **Yifan Sun**[2]
[1] Key Laboratory of Symbolic Computation and Knowledge Engineer, Jilin University, Changchun, China
[2] Baidu Research, China
jialun18@mails.jlu.edu.cn, liwh@jlu.edu.cn

Abstract

This paper considers deep visual recognition on long-tailed data. To make our method general, we tackle two applied scenarios, *i.e.*, deep classification and deep metric learning. Under the long-tailed data distribution, the most classes (*i.e.*, tail classes) only occupy relatively few samples and are prone to lack of within-class diversity. A radical solution is to augment the tail classes with higher diversity. To this end, we introduce a simple and reliable method named Memory-based Jitter (MBJ). We observe that during training, the deep model constantly changes its parameters after every iteration, yielding the phenomenon of *weight jitters*. Consequentially, given a same image as the input, two historical editions of the model generate two different features in the deeply-embedded space, resulting in *feature jitters*. Using a memory bank, we collect these (model or feature) jitters across multiple training iterations and get the so-called Memory-based Jitter. The accumulated jitters enhance the within-class diversity for the tail classes and consequentially improves long-tailed visual recognition. With slight modifications, MBJ is applicable for two fundamental visual recognition tasks, *i.e.*, deep image classification and deep metric learning (on long-tailed data). Extensive experiments on five long-tailed classification benchmarks and two deep metric learning benchmarks demonstrate significant improvement. Moreover, the achieved performance are on par with the state of the art on both tasks.

Introduction

In visual recognition tasks, the long-tailed distribution of the data is a common and natural problem under realistic scenarios (Van Horn and Perona 2017; Lin et al. 2014; Everingham et al. 2010; Guo et al. 2016). A few categories (*i.e.*, the head classes) occupy most of the data while the most categories (*i.e.*, the tail classes) only occupy relatively few data. Such long-tailed distribution significantly challenges deep visual recognition, including both deep image classification (Zhou et al. 2019; Kang et al. 2019; Cao et al. 2019; Van Horn and Perona 2017; Buda, Maki, and Mazurowski 2018a; Cui et al. 2019; Zhu and Yang 2020) and deep metric learning (Liu et al. 2020; Yin et al. 2019; Guo et al. 2016; Thomee et al. 2016). To make our method general, this paper considers long-tailed visual recognition on these two elemental tasks with a uniform motivation.

We recognize the insufficient within-class diversity of the tail classes as the most prominent reason that hinders long-tailed deep visual recognition. In the deeply-embedded feature space, a tail class is under-represented and thus hard to be recognized. To validate this point, we visualize the deep embedding of CIFAR-10 dataset in Fig. 1. When a specified class ("ID-10") degrades from head (Fig.1 (a)) to tail (Fig.1 (b)), its visual concept collapses into a very limited scope in the deep embedding. Consequentially, when we employ the model for inference, samples from "ID-10" may exceed the already-learned scope and are thus easily mis-classified. Intuitively, a radical solution is to augment the tail classes with higher diversity.

We notice two phenomena which are potential for enhancing the tail data diversity, *i.e.*, the weight jitter and the feature jitter. During training, the deep model constantly changes its parameters after every iteration, yielding the phenomenon of *weight jitter*. Consequentially, given a same image as the input, the models at two different iterations generate two different feature representations in the deeply-embedded space, resulting in the phenomenon of *feature jitter*.

Since these jitters are distributed among historical models, we need to accumulate them across multiple training iterations for diversity enhancement. To this end, we employ a memory bank to store the desired jitters, and get the so-called Memory-based Jitter (MBJ). With slight modifications, MBJ is capable to accommodate two elemental visual recognition tasks, *i.e.*, deep image classification and deep metric learning. *On deep image classification*, MBJ collects the historical features (*i.e.*, feature jitters). Consequentially, the feature memory bank accumulates abundant tail-feature jitters, and improves the classification accuracy on tail classes, as shown in Fig. 1(c)). *On deep metric learning*, MBJ collects the weight vectors of the classifier layer instead of the features. Each weight vector is typically viewed as the prototype of a training class, so we name the corresponding memory bank as the prototype memory bank.

Besides the accumulated jitter, MBJ is benefited from a novel re-sampling effect between head and tail classes. On both the classification and the deep metric learning task,

[*]Work done during an internship at Baidu Research.
[†]Corresponding author.

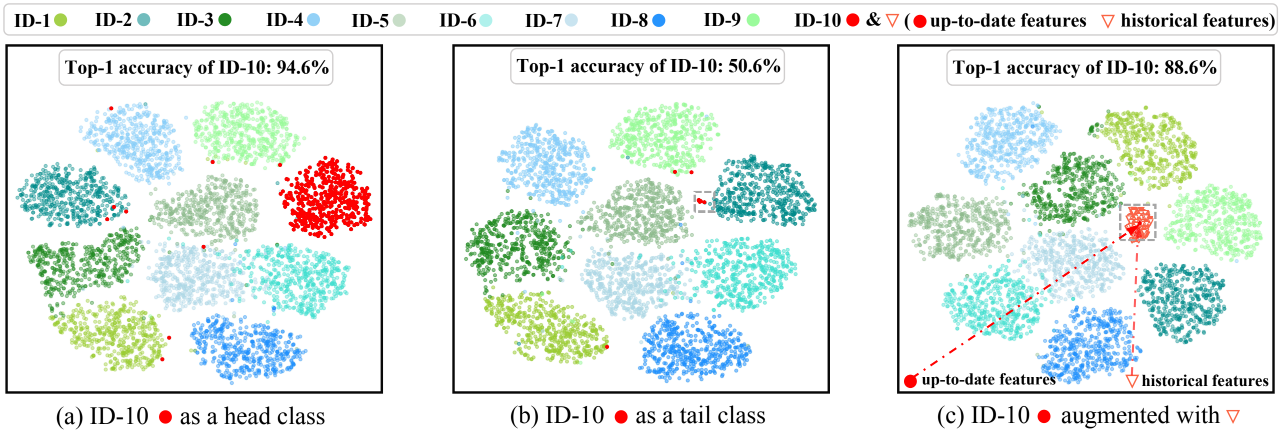

Figure 1: The proposed Memory-based Jitter (MBJ) enhances the tail feature diversity by accumulating historical features into a memory bank. We visualize the feature distribution of CIFAR-10 with t-SNE (Van Der Maaten 2014). We focus on a specified class ID-10. In (a), ID-10 has abundant training samples and is a head class. Its top-1 accuracy is 94.6%. In (b), we reduce the training samples of ID-10, so it becomes a tail class. Due to the lack of within-class diversity, its feature distribution collapses into a very limited scope and the top-1 accuracy dramatically decreases to 50.6%. In (c), MBJ collects historical features distributed among multiple training iterations into a memory bank. The historical features are scattered around the up-to-date features in the deeply-embedded space, yielding the so-called Memory-based Jitter (MBJ). Consequentially, MBJ enhances the tail data diversity and increases the classification accuracy of ID-10 from 50.6% to 88.6%.

MBJ assigns larger sampling rate to the tail classes (than to the head classes). Correspondingly, the tail classes occupy more memory-based jitters than the head classes, which compensates for the imbalanced distribution of the raw data. We note that some recent works (Zhou et al. 2019; Kang et al. 2019) evidence that directly over-sampling the raw images, though alleviates the data imbalance problem to some extent, actually compromises deep embedding learning. In contrast, MBJ maintains the natural sampling frequency on the raw images and re-balances the head and tail classes in the memory bank.

The main contributions of this paper are summarized as follows:

• We find that the weight jitters and the feature jitters are informative clues to gain extra diversity for tail data augmentation.

• We propose Memory-based Jitter to accumulate the jitters within a memory bank and improve deep visual recognition on long-tailed data. MBJ is compatible to two elemental visual tasks, *i.e.*, deep image classification and deep metric learning, with slight modifications.

• MBJ is featured for the memory-based feature space augmentation. It can be treated as a plug-in module and be unified with previous methods.

• We conduct extensive experiments on five classification benchmarks and two metric learning benchmarks (person re-identification, in particular) under long-tailed scenario. On all these benchmarks, we demonstrate the superiority of our methods, which significantly improves the baseline and is on par with the state-of-the-art methods.

Related Work

Re-balancing Strategy

MBJ has a novel re-balancing strategy, compared with prior works on long-tailed visual recognition. Generally, re-balancing aims to highlight the tail classes during training. In prior works, there are two major re-balancing types, *i.e.*, *re-weighting*(Huang et al. 2016; Wang, Ramanan, and Hebert 2017; Cui et al. 2019) and *re-sampling*(Shen, Lin, and Huang 2016; Zhong et al. 2016; Buda, Maki, and Mazurowski 2018b; Byrd and Lipton 2019). Re-weighting strategy allocates larger weights to tail classes in loss function. Re-sampling over-samples the raw images of the tail classes for training.

Different from these prior works, MBJ re-samples the features / prototypes to highlight the tail classes. It thus avoids directly re-sampling the raw data. Since directly re-sampling the raw data actually compromises the deep embedding learning (Zhou et al. 2019; Kang et al. 2019), avoiding such operation substantially benefits MBJ. An ablation study carried out on the long-tailed CIFAR-10 dataset shows that when we remove the jitter augmentation, this novel re-sampling strategy still brings $+2.1\%$ improvement over the baseline. The details are to be accessed in Section .

Memory-based Learning

The memory bank plays a critical role in MBJ. Both the weight jitters and the feature jitters are scattered among sequential training iterations. To accumulate these jitters for tail data augmentation, we employ a memory bank. Since memory-based learning has been explored in several computer vision domains, including unsupervised learning, semi-supervised learning and supervised learning (He et al. 2019; Tarvainen and Valpola 2017; Laine and Aila 2016;

Wang et al. 2019; Santoro et al. 2016; Zhu and Yang 2018), we make a detailed comparison as follows.

In unsupervised learning, (He et al. 2019) employs memory to include more data in the dictionary. It shows that larger optimization scope within a optimization step is beneficial for unsupervised learning (He et al. 2019). In semi-supervised learning, (Laine and Aila 2016; Tarvainen and Valpola 2017) enforce consistency between historical predictions. Such consistency offers auxiliary supervision for the unlabeled data. In supervised deep metric learning, (Wang et al. 2019) uses memory to enhance the hard mining effect. Regardless of their objectives of using memory, they all hold a negative attitude towards the jitters. (He et al. 2019) and (Laine and Aila 2016; Tarvainen and Valpola 2017) suppress the jitters with momentum and consistency constraint, respectively. (Wang et al. 2019) tries to avoid the jitters by delaying the injection of memory.

In contrast to their negative attitude towards jitters, we find that the jitters are informative for long-tailed visual recognition. As a major contribution of this work, we analyze the mechanism in Section and experimentally validate its effectiveness in Section .

Moreover, we notice a recent work IEM (Zhu and Yang 2020) also employs memory for long-tailed image classification. We compare MBJ against IEM in details for clarity. Our method significantly differs from IEM in three aspects, *i.e.*, the applied task, the mechanism and the achieved performance. First, IEM is specified for image classification, while MBJ improves both image classification and deep metric learning with a uniform motivation. Second, IEM considers tail classes are harder to recognize, and thus employ more prototypes from the memory for higher redundancy, while MBJ employs the jitters in memory to augment the diversity of tail data. Finally, on image classification task, MBJ maintains competitive performance with significantly higher computing efficiency. IEM requires extraordinary large amount of memory (up to $50,000$ per class), and achieves Top-1 accuracy of 67% on iNaturalist18. In contrast, MBJ is more memory-efficient and more accuracy. For example, on iNaturalist18 (Van Horn et al. 2018), MBJ only stores $40,000$ memorized features in total and achieves Top-1 accuracy of 66.9% and 70.0% at 90 and 200 epochs, respectively.

With these comparison, we find that MBJ is featured for the memory-based feature augmentation. It is orthogonal to many prior works. Specifically, we note a very recent work RIDE (Wang et al. 2020) using multiple classifiers (experts) ensemble to improve the accuracy of head and tail classes, simultaneously. MBJ can be integrated into RIDE (Wang et al. 2020) for better performance gains.

Proposed Method

Basically, MBJ accumulates historical jitters within a memory bank to enhance the diversity of the tail classes. Under this framework, MBJ adopts slight modifications to accommodate two fundamental visual recognition tasks, *i.e.* feature jitters for image classification and prototype jitters for deep metric learning. Explicitly, deep image classification and deep metric learning have the fundamental differences.

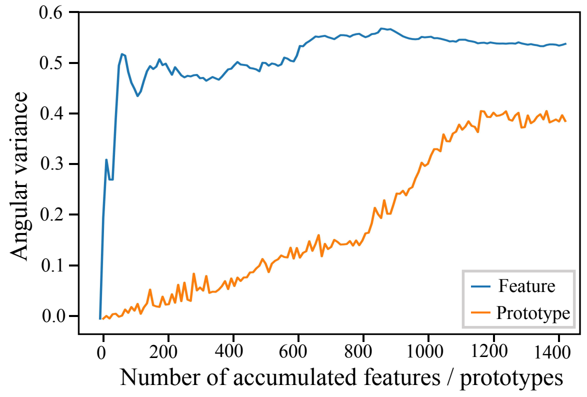

Figure 2: Quantitative statistics on feature jitters and weight jitters. We use a long-tailed CIFAR-10 as the toy dataset and focus on the tail class / image. As the accumulated features / prototypes increase, the angular variance gradually increases.

Specifically, there are two aspects. On one hand, the definitions of two tasks are different. The classification task aims to recognize the already-seen classes. The categories of the training set and testing set are completely overlapped. The deep metric learning task aims to discriminate the unseen classes. The identities of training set and the testing set have no overlap. On the other hand, the optimized objectives of two tasks are also different. In classification task, the model aims to learn an accurate and unbiased classifier that outputs the correct label to the specified instance as much as possible. In metric learning task, the model aims to learn a discriminative feature extractor that encourages the instances from the same class to be closer than those from different classes. In this section, we first analyze the weight jitters and the feature jitters. Then we introduce the MBJ for deep image classification and deep metric learning.

Weight Jitters and Feature Jitters

To illustrate the phenomena of weight jitters and feature jitters, we conduct a toy experiment on CIFAR-10. We set a specified class to contain very limited samples (*i.e.*, 50 samples) so that it turns into a tail class. We train a deep classification model to convergence and then continue the training for observation purpose. Within the following iterations, we record two objects, *i.e.*, 1) the prototype (*i.e.*, the weight vector in the classification layer) of the tail class and 2) the feature of a single tail sample. As the training iterates, both the prototypes and the features accumulate, allowing a quantitative statistic on their variances. We visualize the geometrical angular variance of the accumulated features / prototypes in Fig. 2, from which we draw two observations.

First, we observe considerable variance among the accumulated weight vectors (*i.e.*, prototypes), as well as the accumulated features. It indicates that among multiple iterations, both the prototype of a single class and the feature of a single image keep on changing itself, yielding the phenomena of weight jitters and feature jitters, respectively.

Second, we observe that the above-described jitters re-

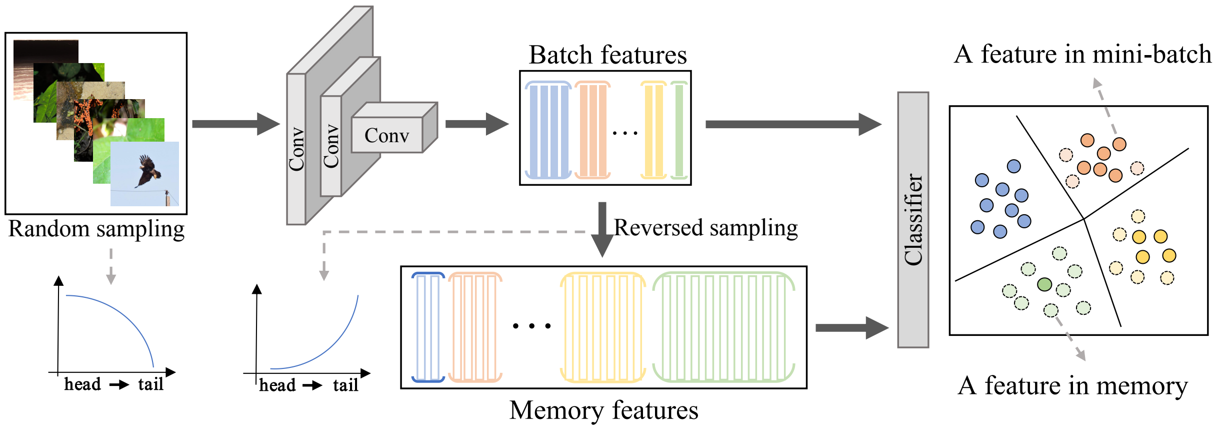

Figure 3: MBJ for deep image classification collects historical features into memory with higher concentration on tail classes. MBJ randomly samples the raw images and transforms them into a batch of features with a convolutional neural network. Given current batch features, MBJ uses a class-specific sampling strategy to collect the features from different classes. Head classes have smaller sampling probability and tail classes have larger sampling probability. These features are stored into a memory bank, *i.e.*, feature memory. MBJ combines the feature memory and the batch features to learn the classifier. The feature memory compensates the tail classes with higher diversity.

quire certain training iterations to accumulate before they reach stable status. When there is only one single feature / prototype, the corresponding variance is naturally 0. As the number of accumulated features / prototypes increases, the variance gradually grows until reaching a stable level.

Based on the above observation, we device MBJ. MBJ uses a memory bank to collect the historical features / prototypes, so as to accumulate the feature / weight jitters for tail augmentation.

MBJ for Deep Image Classification

The pipeline of MBJ for deep image classification is illustrated in Fig.3. In the raw image space, MBJ randomly samples the images without re-balancing the head and tail classes. In each training iteration, the deep model transforms the raw images into a batch of features (from second last layer). Given the features in current mini-batch, MBJ stores them into a memory bank. The memory bank has a larger size than the mini-batch, so it is capable to accumulate historical features across multiple training iterations.

When collecting the features, MBJ lays emphasis on the tail classes, so that the tail and head data will be re-balanced. Specifically, MBJ assigns small sampling probabilities to head classes, as well as relatively large sampling probabilities to tail classes, which is formulated as:

$$P_i = \frac{(1/N_i)^\beta}{\sum_j^C (1/N_j)^\beta} \tag{1}$$

where P_i is the corresponding sampling probability of class i, N_i is the sample number of the i-th class, C is the total number of classes, and β is a hyper-parameter to control the strength of re-balancing. A larger β results in higher priority on accumulating the tail features. We use $\beta = 1.5$ in all of our experiments.

To control the memory size, we use a queue strategy for updating the memory bank. After the memory bank reaches its size limitation, we enqueue the newest features (*i.e.*, the features in current mini-batch), and dequeue the oldest ones.

Given the feature memory and the batch features, MBJ combines both of them to learn the classifier in a joint optimization manner. Specifically, MBJ uses the features in current mini-batch and the weight vectors in the classification layer to deduce a cross-entropy loss, *i.e.*, the loss $\mathcal{L}_{batch}$. Meanwhile, MBJ uses the memorized features and the weight vectors in the classification layer to deduce another cross-entropy loss, *i.e.*, the loss $\mathcal{L}_{memory}$. MBJ sums up those two losses by:

$$\mathcal{L}_{total} = \mathcal{L}_{memory} \times \eta + \mathcal{L}_{batch}, \tag{2}$$

in which η is a weighting factor. The pseudo-code of MBJ for deep image classification task is provided in supplementary material.

MBJ for Deep Metric Learning

A popular baseline (Qian et al. 2019; Liu et al. 2020; Sun et al. 2020; Sohn 2016; Liu et al. 2017; Sun et al. 2017, 2018a) for deep metric learning is as follows: during training, we learn a classification model on the training set. The weight vectors in the classification layer are typically recognized as prototypes for each class. During testing, the distance between two images are measured under the learned deep embedding. Based on this baseline, MBJ collects the historical prototypes into a prototype memory with emphasis on the tail classes.

The sampling strategy is exactly the same as in Eq.1, so we omit the detailed description. Given the historical prototypes in memory and the up-to-date prototypes, MBJ combines both to learn the features. For clarity, we illustrate the learning process with focus on a single feature x under optimization.

To learn with the up-to-date prototypes $W = \{w_1, w_2, \cdots, w_C\}$ (C is the total number of training classes), MBJ adopts a popular deep metric learning method, *i.e.*, CosFace (Wang et al. 2018a), which is formulated as:

$$\mathcal{L}_{batch} = -\log \frac{\exp\left(\alpha(w_y^T x - \delta)\right)}{\exp\left(\alpha(w_y^T x - \delta)\right) + \sum_{k \neq y}^{C} \exp\left(\alpha w_k^T x\right)} \quad (3)$$

in which C is the number of classes, w_y is the prototype of the target class, α is a scale factor and δ is a margin for better similarity separation.

To learn with the prototype memory, MBJ needs to deal with multiple positive prototypes associated with feature x. It is because the weight vector of the target class in the classifier may be sampled at several training iterations. Let us assume there are K positive prototypes $\{u_1, u_2, \cdots, u_K\}$(*i.e.*, weight vectors of the target class), and L negative prototypes $\{v_1, v_2, \cdots, v_L\}$ (*i.e.*, weight vectors of the non-target class). We find that a recent deep metric learning method, *i.e.*, Circle Loss(Sun et al. 2020), allows multiple similarities associated with a single sample feature. Accordingly to Circle Loss, we define the loss function associated with the prototype memory by:

$$\mathcal{L}_{memory} = \log\left[1 + \sum_{j=1}^{L}\sum_{i=1}^{K} \exp(\alpha(v_j^T x - u_i^T x + \delta))\right] \quad (4)$$

Similar to Eq.2, MBJ sums the above two losses (*i.e.*, $\mathcal{L}_{batch}$ and $\mathcal{L}_{memory}$) to optimize the feature x. We note that two editions of MBJ share a unified framework, except for the jitter type. To improve the classification accuracy, MBJ memorizes the feature jitters; To improve the retrieval accuracy, MBJ memorizes the prototype jitters. They have a dual pattern against each other. The pseudo-code of MBJ for deep metric learning task is provided in supplementary material.

Discussions

MBJ is featured for its re-balancing strategy and its augmentation manner. Although re-balancing the features / prototypes (instead of re-balancing the raw data) considerably benefits MBJ (as introduced in Section), we note that the improvement is mainly because the accumulated jitters increase the tail data diversity. Removing the jitters or using other augmentation method significantly compromise MBJ. The details are to be accessed in Section .

Experiments

Datasets and Setup

Deep Classification. Under long-tailed image classification scenario, we evaluate MBJ on 5 datasets, *i.e.*, CIFAR-10, CIFAR-100, ImageNet-LT, Places-LT and iNaturalist18.

For CIFAR datasets, we synthesize several long-tailed version, following (Cao et al. 2019). We use an imbalance ratio to denote the ratio between sample size of the most frequent and least frequent class. Imbalanced ratio (IR) in our experiments are set to 10, 50 and 100, respectively.

ImageNet-LT, Places-LT and iNaturalist18 are publicly available long-tailed dataset. In ImageNet-LT, the maximum of images per class is 1280 and the minimum of images per class is 5. In Places-LT, the largest class has 4980 images while the smallest ones have 5 images. Their test set is balanced. The iNaturalist18 dataset is a large-scale dataset with extremely imbalanced label distribution. It has $437,513$ images from $8,142$ classes. We adopt the official training and validation splits for our experiments.

Deep Metric Learning. We employ the person re-identification (re-ID) (Zhong et al. 2018; Sun et al. 2018b; Zheng et al. 2019) task to evaluate MBJ on deep metric learning. Given a query person, re-ID aims to spot the appearance of the same person in the gallery. The keynote of re-ID is to learn accurate metric that measures the similarity between query and gallery images. We adopt two dataset, *i.e.*, Market-1501(Zheng et al. 2015) and DukeMTMC-reID(Ristani et al. 2016; Zheng, Zheng, and Yang 2017). Following the settings in feature cloud (Liu et al. 2020), we synthesize several long-tailed editions based on the original datasets. For comprehensive evaluation, we vary the number of head classes as 20, 50 and 100, respectively. All the tail classes contain only 5 images per class.

Implementation Details

Parameter Settings. For both task, the re-balancing factor β (Eq. 1) is set to 1.5 and the memory size is set to $5 * C$ (C is the total number of training classes). Please refer to supplementary material for more details.

Experiments on Image Classification

Evaluation on Long-tailed CIFAR-10 / 100 Table 1 compares MBJ with the baseline and several state-of-the-art methods on the long-tailed CIFAR-10 and CIFAR-100. Comparing MBJ with "Basel. (CE)", we find that MBJ significantly improves the baseline. Under the setting of IR 100, for instance, MBJ surpasses the baseline by $+10.6\%$ and $+7.5\%$ top-1 accuracy on CIFAR-10 and CIFAR-100, respectively. Comparing MBJ with several state-of-the-art methods, we find that MBJ is on par with them. For example, comparing MBJ with Hybrid-PSC (Wang et al. 2021), under IR 10, MBJ is slightly lower than it. While, under the more imbalanced case (IR 100 and 50), MBJ is significantly better than it. Especially under the setting of IR 50, MBJ marginally surpasses it by $+2.7\%$ and $+3.7\%$ top-1 accuracy on CIFAR-10 and CIFAR-100, respectively. When we compare MBJ against the recent work RIDE (Wang et al. 2020) and ACE (Cai, Wang, and Hwang 2021), MBJ is lower than RIDE (Wang et al. 2020) and ACE (Cai, Wang, and Hwang 2021) in some cases. It is because they use the multiple experts ensemble. MBJ is featured of the memory-based feature augmentation, and thus it can be integrated into them, achieving the better performance. Specifically, we combine MBJ with RIDE (Wang et al. 2020). "MBJ + RIDE" achieves the further improvement against RIDE (Wang et al. 2020), and is better than ACE (Cai, Wang, and Hwang 2021). More methods combined with MBJ are provided in supplementary material.

Evaluation on ImageNet-LT and Places-LT The experiment results on ImageNet-LT and Places-LT are shown in Table 2. These two datasets offer separate evaluations under Many-shot (more than 100 training images per class), Medium-shot (20 to 100 training images per class), Few-shot (less than 20 images per class) and the Overall per-

Dataset	Long-tailed CIFAR-10			Long-tailed CIFAR-100		
Imbalanced ratio (IR)	100	50	10	100	50	10
Basel. (CE)	70.4	74.8	86.4	38.3	43.9	55.7
LDAM-DRW (Cao et al. 2019)	77.0	81.0	88.2	42.0	46.6	58.7
BBN (Zhou et al. 2019)	79.8	82.2	88.3	43.7	47.0	59.1
SSP (Yang and Xu 2020)	77.8	82.1	88.5	43.4	47.1	58.9
De-confound-TDE (Tang, Huang, and Zhang 2020)	80.6	83.6	88.5	44.1	50.3	59.6
Hybrid-PSC (Wang et al. 2021)	78.8	83.9	**90.1**	45.0	48.9	**62.7**
MetaSAug (Li et al. 2021)	80.5	80.0	89.4	46.9	51.9	61.7
LADE (Hong et al. 2021)	-	-	-	45.4	50.5	61.7
DiVE (He, Wu, and Wei 2021)	-	-	-	45.4	51.1	62.0
RIDE (4 experts)† (Wang et al. 2020)	81.1	87.0	88.3	48.7	59.0	58.4
ACE (3 experts) (Cai, Wang, and Hwang 2021)	81.2	84.3	-	49.4	50.7	-
MBJ	81.0	86.6	88.8	45.8	52.6	60.7
MBJ + RIDE (4 experts)	**82.2**	**87.3**	89.4	**49.9**	**59.8**	62.1

Table 1: Comparison with baseline and the state-of-the-art methods on long-tailed CIFAR-10 and CIFAR-100. We report top-1 accuracy rates. The best results are in bold. † denotes our reproduced results with released code.

Methods	ImageNet-LT				Places-LT			
	Many	Medium	Few	Overall	Many	Medium	Few	Overall
Basel.(CE)	65.9	37.5	7.7	44.4	45.7	27.3	8.2	30.2
Decouple-LWS (Kang et al. 2019)	60.2	47.2	30.3	49.9	40.6	39.1	28.6	37.6
FSA (Chu et al. 2020)	-	-	-	-	42.8	37.5	22.7	36.4
De-confound-TDE (Tang, Huang, and Zhang 2020)	62.7	48.8	31.6	51.8	-	-	-	-
DisAlign (Zhang et al. 2021)	61.5	50.7	33.1	52.6	40.4	42.4	30.1	39.3
LADE (Hong et al. 2021)	62.3	49.3	31.2	51.9	42.8	39.0	31.2	38.8
DiVE (He, Wu, and Wei 2021)	64.1	50.5	31.5	53.1	42.8	39.0	31.2	38.8
RIDE (4 experts)† (Wang et al. 2020)	67.8	53.4	36.2	56.6	46.1	43.3	32.3	41.2
MBJ	61.6	48.4	**39.0**	52.1	39.5	38.2	35.5	38.1
MBJ + RIDE (4 experts)	**68.4**	**54.1**	37.7	**57.7**	**46.6**	**44.8**	**37.2**	**42.5**

Table 2: Comparison with baseline and the state-of-the-art methods on long-tailed CIFAR-10 and CIFAR-100. We report top-1 accuracy rates. The best results are in bold. † denotes our reproduced results with released code.

Methods	90 E	200 E
Basel.(CE)	61.1	65.3
IEM* (Zhu and Yang 2020)	67.0	-
LDAM-DRW † (Cao et al. 2019)	64.6	66.1
BBN (Zhou et al. 2019)	66.3	69.7
Decouple-LWS (Kang et al. 2019)	65.9	69.5
MetaSAug (Li et al. 2021)	-	68.9
Hybrid-PSC (Wang et al. 2021)	68.1	70.4
DisAlign (Zhang et al. 2021)	67.8	70.6
LADE (Hong et al. 2021)	-	70.0
DiVE (He, Wu, and Wei 2021)	**69.1**	71.7
RIDE (4 experts)† (Wang et al. 2020)	-	72.6
ACE (3 experts) (Cai, Wang, and Hwang 2021)	-	72.6
MBJ	66.9	70.0
MBJ + RIDE (4 experts)	-	**73.2**

Table 3: Top-1 accuracy on iNaturalist18. All models use the ResNet-50 (He et al. 2016) backbone. IEM* denotes the IEM (Zhu and Yang 2020) using global feature for fair comparison. † denotes our reproduced results with released code. The best results are in bold.

formance, respectively. From Table 2, We draw three observations as follows: First, under Many-shot and Medium-shot, MBJ achieves comparable accuracy. For example, on ImageNet-LT, MBJ is slightly lower than LADE (Hong et al. 2021) by -0.7% (Many-shot) and -0.9% (Medium-shot). Second, under Few-shot, MBJ exhibits significant superiority against all the competing methods. On Places-LT, MBJ surpasses the second best method RIDE (Wang et al. 2020) by $+3.2\%$ top-1 accuracy. Third, due to significant superiority under the Few-shot, as well as the comparable performance under Many-shot and Medium-shot, the Overall performance of MBJ is on par with the state of the art.

Moreover, we notice that most the methods (including MBJ) actually lose some accuracy under the Many-shot, compared with the baseline. Only RIDE (Wang et al. 2020) improves the accuracy of Many-shot, Medium-shot and Few-shot, simultaneously, because RIDE (Wang et al. 2020) uses the multiple experts ensemble. We combine MBJ with RIDE, then we further improve the performance.

Evaluation on INaturalist18 We further evaluate MBJ on the large-scale long-tailed dataset *i.e.* iNaturalist18. The results are shown in Table 3. For fair comparison, we report performance achieved at 90 and 200 training epochs, following the common practice of previous works. Under both settings, MBJ achieves competitive accuracy. When MBJ is combined with RIDE (Wang et al. 2020), "MBJ + RIDE" achieves the best performance.

Experiments on Deep Metric Learning

We evaluate MBJ under a popular deep metric learning task (*i.e.*, re-ID). We note that the long-tail problem on this task has been noticed recently, and the competing methods are

Method	Market-1501						DukeMTMC-reID					
	H100		H50		H20		H100		H50		H20	
	mAP	R-1	mAP	R-1	mAP	R-1	mAP	R-1	mAP	R-1	mAP	R-1
Baseline	62.8	83.8	60.5	80.7	55.6	78.6	52.6	70.3	48.0	67.7	47.0	66.0
Feature cloud (Liu et al. 2020)	68.7	86.5	67.3	84.9	64.1	83.2	55.6	74.8	53.1	73.0	52.4	72.7
MBJ	**72.6**	**88.4**	**68.8**	**86.2**	**66.7**	**84.8**	**60.8**	**78.6**	**56.7**	**74.4**	**57.9**	**75.5**

Table 4: Evaluation of MBJ on long-tailed re-ID task. Under each dataset, there are three different long-tailed conditions, *i.e.*, "H100": 100 head classes. "H50": 50 head classes. "H20": 20 head classes. All the tail classes contain only 5 images per class. We report Rank-1 accuracy (R-1) and mAP on Market-1501 and DukeMTMC-reID. Best performance are in bold.

relatively few. Table 4 compares MBJ with re-ID baseline (CosFace (Wang et al. 2018b)) and a state-of-the-art method (Feature Cloud (Liu et al. 2020)), from which we draw two observations:

First, under typical long-tailed distribution, MBJ significantly improves the re-ID baseline. When there are only 20 head classes ("H20"), MBJ achieves $+11.1\%$ and $+10.9\%$ mAP on Market-1501 and DukeMTMC-reID, respectively.

Second, MBJ marginally surpasses the recent state-of-the-art, *i.e.*, Feature Cloud (Liu et al. 2020). For example, under three long-tailed condition on Market-1501, MBJ achieves 72.6%, 68.8% and 66.7% mAP, which are higher than Feature Cloud by $+3.9\%$, $+1.5\%$ and $+2.6\%$, respectively. MBJ obtain the new state-of-the-art performance.

Ablation Study

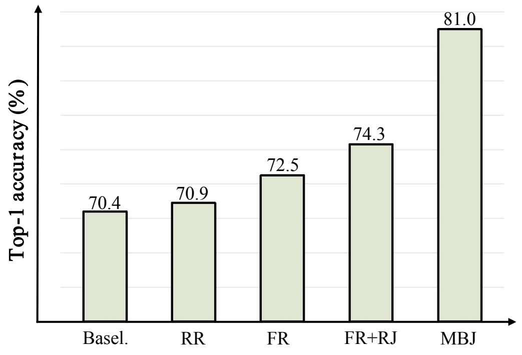

Figure 4: Ablation study on the effect of re-balancing strategy and jitters. "Basel." is the baseline without any re-balancing or tail augmentation. "RR" re-balances the raw images. "FR" adopts the feature re-balancing in MBJ and removes the feature jitters. "FR+RJ" replaces the feature jitters with Gaussian-distributed disturbance (for tail augmentation). The proposed MBJ marginally surpasses the baseline and all the other counterparts. It indicates that the jitter effect is the major reason for the superiority of MBJ.

Decoupling Re-balancing and Jitters When MBJ collects jitters into the memory bank, it has two coupling effects, *i.e.*, a) re-balancing the head and tail distribution and b) accumulating the jitters. On the long-tailed CIFAR-10 (IR 100), we design an ablation study to decouple those two effects. Specifically, we train three different models as follows: 1) Model RR re-balances the raw images by over-sampling the tail classes. 2) Model FR re-balances the feature by over-sampling the tail features in current mini-batch. However, it does NOT collect historical features into the memory bank. In another word, Variant A maintains the re-balancing strategy of MBJ and removes the jitters. 3) Model FR+RJ over-samples the tail features and augments them with Gaussian-distributed disturbance. The three models are built upon on a same converged model and have the same loss items and re-sampling factor β as MBJ.

Fig. 4 compares these three models with the baseline and MBJ, from which we draw two observations:First, comparing "FR" against "RR" and "Basel.", we find that re-balancing the features considerably benefits MBJ. Specifically, directly re-balancing the raw data actually brings no obvious improvement over the baseline. It is consistent with the observation in LDAM (Cao et al. 2019). According to (Cao et al. 2019; Zhou et al. 2019), it is because directly re-sampling the raw data compromises the deep embedding learning. In contrast, re-balancing the features avoids deterioration on the deep embedding and considerably increases the top-1 accuracy by $+2.1\%$. Second, comparing "MBJ" against "FR" and "FR+RJ", we find that the accumulated jitters is the dominating reason for the superiority of MBJ. While re-balancing the feature (FR) improves the baseline by $+2.1\%$ top-1 accuracy, accumulating jitters (MBJ) further brings a much larger improvement of $+8.5\%$ accuracy. Moreover, though adding random disturbance ("FR+RJ") does obtain some degree of feature augmentation as well, its improvement is very limited and is much inferior to the proposed MBJ.

Conclusion

This paper proposes Memory-based Jitter (MBJ) to improve long-tailed visual recognition under both deep classification and deep metric learning tasks. The insights behind MBJ are two-fold. First, during training a deep model, the weight vectors and the features keep on changing after each iteration, resulting in the phenomena of (weight and feature) jitters. Second, accumulating these jitters provides extra augmentation for the tail data. Experimental results confirm MBJ the effectiveness of MBJ. MBJ can be treated as a plug-in module and be unified with the previous achieving the state-of-the-art performance on both deep image classification and deep metric learning.

An interesting observation is that MBJ favors different types of memory, depending on the specified task. For deep image classification, it favors the feature memory, while for deep metric learning, it favors the prototype memory.

References

Buda, M.; Maki, A.; and Mazurowski, M. A. 2018a. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106: 249–259.

Buda, M.; Maki, A.; and Mazurowski, M. A. 2018b. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106: 249–259.

Byrd, J.; and Lipton, Z. 2019. What is the Effect of Importance Weighting in Deep Learning? In *International Conference on Machine Learning*, 872–881.

Cai, J.; Wang, Y.; and Hwang, J.-N. 2021. ACE: Ally Complementary Experts for Solving Long-Tailed Recognition in One-Shot. *arXiv preprint arXiv:2108.02385*.

Cao, K.; Wei, C.; Gaidon, A.; Arechiga, N.; and Ma, T. 2019. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, 1565–1576.

Chu, P.; Bian, X.; Liu, S.; and Ling, H. 2020. Feature Space Augmentation for Long-Tailed Data. In *European Conf. on Computer Vision (ECCV)*.

Cui, Y.; Jia, M.; Lin, T.-Y.; Song, Y.; and Belongie, S. 2019. Class-balanced loss based on effective number of samples. In *CVPR*, 9268–9277.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338.

Guo, Y.; Zhang, L.; Hu, Y.; He, X.; and Gao, J. 2016. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European conference on computer vision*, 87–102. Springer.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2019. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, Y.-Y.; Wu, J.; and Wei, X.-S. 2021. Distilling Virtual Examples for Long-tailed Recognition. *arXiv preprint arXiv:2103.15042*.

Hong, Y.; Han, S.; Choi, K.; Seo, S.; Kim, B.; and Chang, B. 2021. Disentangling Label Distribution for Long-tailed Visual Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6626–6636.

Huang, C.; Li, Y.; Change Loy, C.; and Tang, X. 2016. Learning deep representation for imbalanced classification. In *CVPR*, 5375–5384.

Kang, B.; Xie, S.; Rohrbach, M.; Yan, Z.; Gordo, A.; Feng, J.; and Kalantidis, Y. 2019. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*.

Laine, S.; and Aila, T. 2016. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*.

Li, S.; Gong, K.; Liu, C. H.; Wang, Y.; Qiao, F.; and Cheng, X. 2021. MetaSAug: Meta Semantic Augmentation for Long-Tailed Visual Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5212–5221.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.

Liu, J.; Sun, Y.; Han, C.; Dou, Z.; and Li, W. 2020. Deep Representation Learning on Long-tailed Data: A Learnable Embedding Augmentation Perspective. *arXiv preprint arXiv:2002.10826*.

Liu, X.; Vijaya Kumar, B. V. K.; You, J.; and Jia, P. 2017. Adaptive Deep Metric Learning for Identity-Aware Facial Expression Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Qian, Q.; Shang, L.; Sun, B.; Hu, J.; Li, H.; and Jin, R. 2019. SoftTriple Loss: Deep Metric Learning Without Triplet Sampling. In *Proceedings of the IEEE International Conference on Computer Vision*, 6450–6458.

Ristani, E.; Solera, F.; Zou, R.; Cucchiara, R.; and Tomasi, C. 2016. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, 17–35. Springer.

Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; and Lillicrap, T. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, 1842–1850.

Shen, L.; Lin, Z.; and Huang, Q. 2016. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 467–482.

Sohn, K. 2016. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*, 1857–1865. Curran Associates, Inc.

Sun, Y.; Cheng, C.; Zhang, Y.; Zhang, C.; Zheng, L.; Wang, Z.; and Wei, Y. 2020. Circle loss: A unified perspective of pair similarity optimization. *arXiv preprint arXiv:2002.10857*.

Sun, Y.; Zheng, L.; Deng, W.; and Wang, S. 2017. Svdnet for pedestrian retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, 3800–3808.

Sun, Y.; Zheng, L.; Yang, Y.; Tian, Q.; and Wang, S. 2018a. Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline). In *Proceedings of the European Conference on Computer Vision (ECCV)*, 480–496.

Sun, Y.; Zheng, L.; Yang, Y.; Tian, Q.; and Wang, S. 2018b. Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline). In *Proceedings of the European Conference on Computer Vision (ECCV)*, 480–496.

Tang, K.; Huang, J.; and Zhang, H. 2020. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *arXiv preprint arXiv:2009.12991*.

Tarvainen, A.; and Valpola, H. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, 1195–1204.

Thomee, B.; Shamma, D. A.; Friedland, G.; Elizalde, B.; Ni, K.; Poland, D.; Borth, D.; and Li, L.-J. 2016. YFCC100M: The New Data in Multimedia Research. *Commun. ACM*, 59(2): 64–73.

Van Der Maaten, L. 2014. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1): 3221–3245.

Van Horn, G.; Mac Aodha, O.; Song, Y.; Cui, Y.; Sun, C.; Shepard, A.; Adam, H.; Perona, P.; and Belongie, S. 2018. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8769–8778.

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

Wang, H.; Wang, Y.; Zhou, Z.; Ji, X.; Gong, D.; Zhou, J.; Li, Z.; and Liu, W. 2018a. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5265–5274.

Wang, H.; Wang, Y.; Zhou, Z.; Ji, X.; Gong, D.; Zhou, J.; Li, Z.; and Liu, W. 2018b. Cosface: Large margin cosine loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5265–5274.

Wang, P.; Han, K.; Wei, X.-S.; Zhang, L.; and Wang, L. 2021. Contrastive Learning based Hybrid Networks for Long-Tailed Image Classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 943–952.

Wang, X.; Lian, L.; Miao, Z.; Liu, Z.; and Yu, S. X. 2020. Long-tailed Recognition by Routing Diverse Distribution-Aware Experts. *arXiv preprint arXiv:2010.01809*.

Wang, X.; Zhang, H.; Huang, W.; and Scott, M. R. 2019. Cross-Batch Memory for Embedding Learning. *arXiv preprint arXiv:1912.06798*.

Wang, Y.-X.; Ramanan, D.; and Hebert, M. 2017. Learning to model the tail. In *NeurIPS*, 7029–7039.

Yang, Y.; and Xu, Z. 2020. Rethinking the Value of Labels for Improving Class-Imbalanced Learning. *arXiv preprint arXiv:2006.07529*.

Yin, X.; Yu, X.; Sohn, K.; Liu, X.; and Chandraker, M. 2019. Feature transfer learning for face recognition with under-represented data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5704–5713.

Zhang, S.; Li, Z.; Yan, S.; He, X.; and Sun, J. 2021. Distribution Alignment: A Unified Framework for Long-tail Visual Recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2361–2370.

Zheng, L.; Shen, L.; Tian, L.; Wang, S.; Wang, J.; and Tian, Q. 2015. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, 1116–1124.

Zheng, Z.; Yang, X.; Yu, Z.; Zheng, L.; Yang, Y.; and Kautz, J. 2019. Joint discriminative and generative learning for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2138–2147.

Zheng, Z.; Zheng, L.; and Yang, Y. 2017. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In *Proceedings of the IEEE International Conference on Computer Vision*, 3754–3762.

Zhong, Q.; Li, C.; Zhang, Y.; Sun, H.; Yang, S.; Xie, D.; and Pu, S. 2016. Towards good practices for recognition & detection. In *CVPR workshops*, volume 1.

Zhong, Z.; Zheng, L.; Zheng, Z.; Li, S.; and Yang, Y. 2018. Camstyle: A novel data augmentation method for person re-identification. *IEEE Transactions on Image Processing*, 28(3): 1176–1190.

Zhou, B.; Cui, Q.; Wei, X.-S.; and Chen, Z.-M. 2019. BBN: Bilateral-Branch Network with Cumulative Learning for Long-Tailed Visual Recognition. *arXiv preprint arXiv:1912.02413*.

Zhu, L.; and Yang, Y. 2018. Compound memory networks for few-shot video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 751–766.

Zhu, L.; and Yang, Y. 2020. 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*, 4344–4353.