Long-Tailed Recognition via Weight Balancing

Shaden Alshammari[*]   Yu-Xiong Wang[♯]   Deva Ramanan[♯,†]   Shu Kong[♯]

[*]MIT   [♯]UIUC   [†]Argo AI   [♯]CMU

shaden@mit.edu   yxw@illinois.edu   {deva, shuk}@andrew.cmu.edu

https://github.com/ShadeAlsha/LTR-weight-balancing

(a) per-class classification accuracy vs. class cardinality on CIFAR100-LT (imalance factor 100)

(b) norms of per-class weights from the learned classifier vs. class cardinality
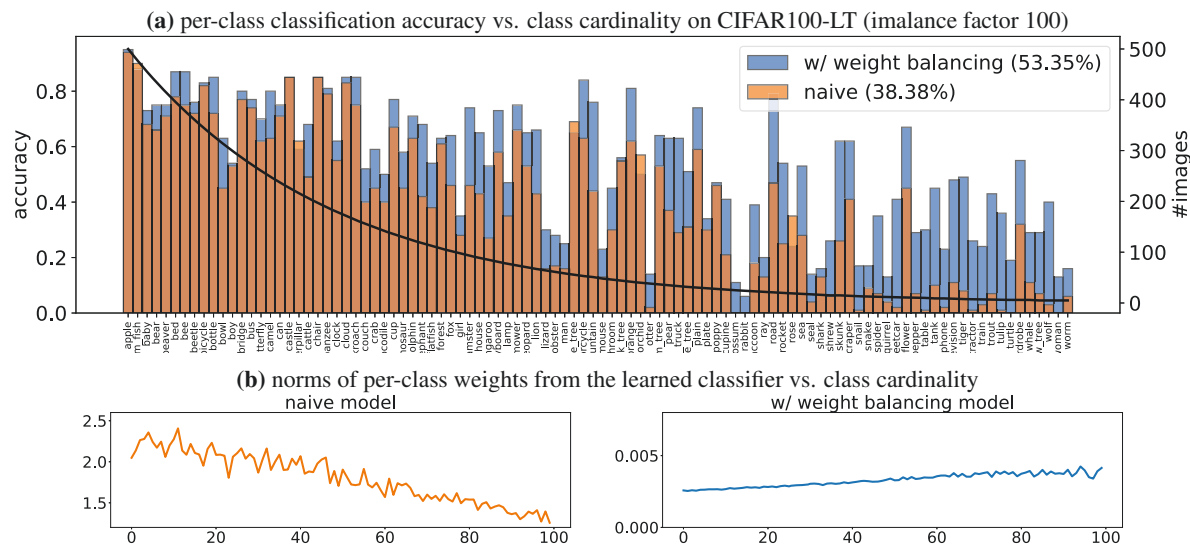
Figure 1. Long-tailed recognition (LTR) requires training on long-tailed class distributed data (black curve in **(a)**). **(a)** Networks naively trained on such data are biased toward common classes in terms of higher accuracy (orange bars). **(b)**-left plots the L2 norms of per-class weights in the naive classifier. Clearly, classifiers of common classes have "artificially" larger norms than rare ones, because they are trained with more data. This can result in over-predictions of common classes, or alternatively, under-predictions of rare classes. This observation motivates us to balance norms via *parameter regularization*. To do so, we explore simple weight balancing techniques including L2-normalization, weight decay, and the MaxNorm constraint. We find that applying the latter two results in class weights to be far more balanced (**(b)**-right), allowing rare classes to have a "fighting chance" when competing with common classes. Our model boosts overall accuracy to 53.35% (blue bars in **(a)**), significantly higher than the naive model (38.38%) and prior art, e.g., RIDE (49.1%) [73], ACE (49.6%) [10], and PaCo (52.0%) [17]. Results are from experiments (Table 1) on CIFAR100-LT with an imbalance factor 100 [12].

Abstract

In the real open world, data tends to follow long-tailed class distributions, motivating the well-studied long-tailed recognition (LTR) problem. Naive training produces models that are biased toward common classes in terms of higher accuracy. The key to addressing LTR is to balance various aspects including data distribution, training losses, and gradients in learning. We explore an orthogonal direction, **weight balancing***, motivated by the empirical observation that the naively trained classifier has "artificially" larger weights in norm for common classes (because there exists abundant data to train them, unlike the rare classes). We investigate three techniques to balance weights, L2-normalization, weight decay, and MaxNorm. We first point out that L2-normalization "perfectly" balances per-class weights to be unit norm, but such a hard constraint might prevent classes from learning better classifiers. In contrast, weight decay penalizes larger weights more heavily and so learns small balanced weights; the MaxNorm constraint encourages growing small weights within a norm ball but caps all the weights by the radius. Our extensive study shows that both help learn balanced weights and greatly improve the LTR accuracy. Surprisingly, weight decay, although underexplored in LTR, significantly improves over prior work. Therefore, we adopt a two-stage training paradigm and propose a simple approach to LTR: (1) learning features using the cross-entropy loss by tuning weight decay, and (2) learning classifiers using class-balanced loss by tuning weight decay and MaxNorm. Our approach achieves the state-of-the-art accuracy on five standard benchmarks, serving as a future baseline for long-tailed recognition.*

1. Introduction

In the real open world, data tends to follow long-tailed distributions [8, 60, 84, 85]. Through the lens of classification, this means that the number of per-class data, or class cardinality, is heavily imbalanced [27, 72]. Numerous applications emphasize the rare classes. For example, autonomous vehicles should recognize not only common objects such as cars and pedestrians, but also rare ones like strollers and animals for driving safety [41]. A bio-image analysis system should recognize both commonly- and rarely-seen species for ecological research [63, 72]. This motivates the well-studied problem of long-tailed recognition (LTR), which trains on class-imbalanced data and aims to achieve high accuracy averaged across all the classes [84]. LTR has attracted increasing attention especially using deep neural networks [12, 38, 78].

Status quo. Because common classes have significantly more training data than rare classes, they dominate the training loss, contribute the most of gradients, and obtain high accuracy [84]. Consequently, a naively trained model performs well on them but significantly worse on the rare classes (Fig. 1a). The key to addressing LTR is to balance various aspects. Many methods propose to balance per-class data distributions during training by upsampling rare classes or downsampling common classes [14, 22, 23]. Some others balance the losses or gradients during training [12, 19, 40, 71]. Some approaches adopt transfer learning that learn features on common classes and use the features to learn rare-class classifiers [37, 49, 74, 87]. It shows that decoupling feature learning and classifier learning leads to significant improvement over models that train them jointly [38]. From benchmarking results, the state-of-the-art accuracy is achieved by either ensembling expert models [10, 23, 26, 73, 76] or the adoption of self-supervised pre-training with aggressive data augmentation techniques [17].

Motivation. We observe that a naively trained model on long-tailed class distributed data has "artificially" large weights for common classes (Fig 1b). Prior work also notes this observation [38]. Intuitively, this is because common classes have more training data that significantly grows classifier weights (Fig. 2a). This motivates our work to balance network weights across classes for long-tailed recognition. In contrast to existing methods (as exhaustively reviewed in a recent survey paper [84]), our work explores an orthogonal direction of *weight balancing*.

Contribution. To balance network weights in norm, we study three simple techniques. We first point out that *L2-normalization* perfectly balances classifier weights to have unit norm (Fig. 2b). However, L2-normalization might be too strict to learn flexible parameters for better classifiers. We then study *weight decay* [29, 44] and the *MaxNorm* constraint [35, 66]. Weight decay penalizes larger weights more heavily and so learns small balanced weights (Fig. 2c);

MaxNorm encourages growing small weights within a norm ball and caps all the weights by the radius (Fig. 2d). We find that both effectively learn balanced weights and boost LTR performance, although these well-known regularizers are *underexplored* in the LTR literature. Please refer to Fig. 1 for a nutshell of our work.

Key Findings. We show how simple regularizers boost LTR performance. Without inventing new losses or adopting aggressive augmentation techniques or designing new network modules, we follow the simple two-stage training paradigm [38] and derive a simple approach that rivals or outperforms the state-of-the-art methods: (1) train a backbone using the standard cross-entropy loss by properly tuning *weight decay*, and (2) train the classifier using a class-balanced loss by tuning *weight decay* and *MaxNorm*. It is important to note how our simple approach challenges the increasingly complicated LTR models, and hence serves as a strong future baseline for LTR.

2. Related Work

Long-Tailed Recognition (LTR). Real-world data tends to follow long-tailed class distributions, i.e., a few classes are commonly seen that have significantly more data than many classes that are infrequently / rarely seen. As a result, a model naively trained on such data performs significantly worse on rare classes than common classes. LTR requires training on such data to achieve high accuracy averaged across all classes [12, 38, 78]. For LTR, numerous methods emphasize the accuracy on rare-classes. *Data rebalancing* techniques resample the training data to achieve a more balanced data distribution across classes [53, 67], such as over-sampling rare-classes [14, 28] and undersampling common-classes [21]. *Class-balanced loss reweighting* assigns weights to the classes [12, 19, 36, 39, 40, 83], or even training examples [39, 47, 62, 68], aiming to modify their gradients to make the class-imbalanced data contribute properly to training. *Transfer learning* methods transfer feature representations learned on the common-classes to the rare-classes [48, 79]. Recent work examines the training procedure and finds LTR to be better addressed by decoupling feature learning and classifier learning, rather than training them jointly [38, 88]. It is found that the SGD momentum causes issues in LTR that prevent further improvement [71] . Other sophisticated methods exploit self-supervised pretraining with more aggressive data augmentation techniques [17], or ensemble expert models trained on different data regimes [10, 73]. For a comprehensive review of the LTR literature, we refer the reader to the recent survey paper [84]. Different from all the existing methods, we explore an orthogonal direction of *parameter regularization*, leading to a much simpler approach to LTR.

Parameter Regularization adds extra information to solve an ill-posed problem, improving generalizability and
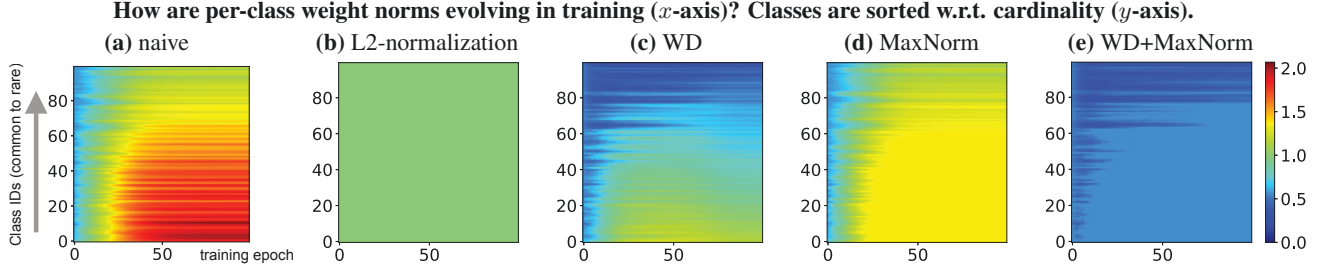
How are per-class weight norms evolving in training (x-axis)? Classes are sorted w.r.t. cardinality (y-axis).

Figure 2. **(a)** In a naive model, all weights grow in norm during training, while those of common classes grow much faster. **(b)** Because L2-normalization constrains weights to be unit-norm, weight norms stay constant during training. **(c)** Weight decay (WD) regularizes all weights to be small while still allowing them to grow. **(d)** MaxNorm caps large weights (of common classes) while letting small weights grow. **(e)** Combining weight decay and MaxNorm results in small and balanced weights in norm. All plots share the same color map.

preventing overfitting [7, 9, 57]. Regularization plays a crucial role in deep learning [43]. One well-known regularization is weight decay, which often applies L2-norm penalty on network weights [29, 44, 52]. There exist many more regularizations [24, 45], such as weight normalization [1, 64], MaxNorm constraints [25, 35, 66], data augmentation [82], and dropout [35]. In this work, we particularly examine the well-known yet *underexplored* regularizers in the LTR literature: L2-normalization, weight decay, and the MaxNorm constraint [25, 35, 66].

Stage-wise Training turns to be effective in training deep networks [31, 50, 80, 86]. This can date back to stage-wise layer pretraining [4, 34]. Recently, Kang et al. convincingly demonstrate that stage-wise training is important to LTR [38]. Concretely, Kang et al. propose to decouple feature learning and classifier learning into two independent stages [38]: (1) feature learning using the standard cross-entropy loss, and (2) classifier learning over the learned feature using a class-balancing loss. While they did not explain why a single one-stage training with the class-balancing loss performed poorly, intuitively, this is because a class-balancing loss artificially scales up gradients computed from rare-class training data, which hurts the feature representation learning and hence the final LTR performance. Follow-up work indirectly demonstrates this intuition with improved performance by stabilizing gradients during training [61, 71]. In our paper, we adopt this two-stage training procedure, but focus on how to balance network weights for LTR.

3. Weight Balancing for Long-Tailed Learning

Preliminaries. Long-tailed recognition (LTR) aims to train over a training set $\mathcal{D}=\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where data example $\mathbf{x}_i$ is labeled as $y_i \in [1, \ldots, K]$. For class-k, $\mathcal{D}_k$ is the set of all its training examples and $|\mathcal{D}_k|$ is its cardinality. The imbalance factor, $\text{IF}=\frac{\max_k |\mathcal{D}_k|}{\min_k |\mathcal{D}_k|}$, measures how imbalanced the long-tailed training set is. For LTR, IF$\gg$1. LTR emphasizes classification accuracy averaged over classes, i.e., accuracy$=\frac{1}{K}\sum_k \text{acc}_k$, where acc$_k$ is the accuracy com-

puted over testing examples of class-k.

LTR focuses on learning a K-way classification network $f(\cdot; \boldsymbol{\Theta})$ parameterized by $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_{l,j}\}$, where $\boldsymbol{\theta}_{l,j}$ is the jth filter weights at layer-l. In a conv-layer, $\boldsymbol{\theta}_{l,j}$ is a 3D kernel that convolves the input (activation). For brevity, we denote $\boldsymbol{\theta}_k$ as the classifier filter corresponding to class-k. Given a data example $\mathbf{x}_i$, the network predicts a label $y_i' = f(\mathbf{x}_i; \boldsymbol{\Theta})$. We measure the prediction error between y_i' and the ground-truth y_i using a cost function $\ell(y_i', y_i)$, e.g., a cross-entropy (CE) loss [7, 56] or a class-balanced loss (CB) [19]. To train the network $f(\cdot; \boldsymbol{\Theta})$, we optimize $\boldsymbol{\Theta}$ by minimizing $\ell(y_i', y_i)$ over the whole training set $\mathcal{D}$:

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} F(\boldsymbol{\Theta}; \mathcal{D}) \equiv \sum_{i=1}^N \ell\big(f(\mathbf{x}_i; \boldsymbol{\Theta}), y_i\big). \quad (1)$$

Naively solving (1) produces a classifier (i.e., the last layer) that has artificially large weights in norm for common classes (Fig. 1b-left, Fig. 2a). Therefore, we are motivated to learn a balanced classifier by regularizing classifier weights, denoted by $\boldsymbol{\theta}_k$ for $k = 1, \ldots, K$. Intermediate layers also have imbalanced filter weights (Fig. 3) even though a filter tends to fire on multiple classes [2, 81]. Generally, one can also balance the weights at intermediate layers, and our study shows that doing so boosts performance. Nevertheless, to simplify presentation in the following, we focus on regularization on the classifier weights $\boldsymbol{\theta}_k$'s.

3.1. Weight Balancing Techniques

We examine the following three techniques to balance weights with respect to norms.

L2-normalization. A "perfect" way to balance the classifier weights $\boldsymbol{\theta}_k$'s is to L2-normalize the classifier weights:

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} F(\boldsymbol{\Theta}; \mathcal{D}), \quad s.t. \ \ \|\boldsymbol{\theta}_k\|_2^2 = 1, \ \forall k. \quad (2)$$

As L2-normalization forces weights to be unit-length, the classifier weights will have unit norm constant during training (Fig. 2b). Inspired by [38], we also post-hoc L2-normalize a trained classifier, i.e., $\boldsymbol{\theta}_k' = \boldsymbol{\theta}_k / \|\boldsymbol{\theta}_k\|_2$. We find

that post-hoc L2-normalization oftentimes improves LTR performance, favoring rare-classes yet sacrificing common-class accuracy. But it can also significantly decrease overall performance, e.g., on iNaturalist in Table 3. Post-hoc L2-normalization is similar to the τ-normalization [38], which allows varied per-class weight norms (rather than forcing them to be the same) and achieves better LTR performance. This suggests that L2-normalization is too strict to strike a balance among the long-tailed distributed classes. Importantly, our exploration finds that, while training with an L2-normalization constraint on the classifier improves over naive training, it underperforms the other two regularizers described below.

Weight Decay is a well-studied technique [44, 55] used to constrain a network by limiting the growth of the network weights. It decreases the complexity of the network, effectively mitigating overfitting and improving generalization. Weight decay typically applies an L2-norm penalty to the network weights (we focus on the classifier θ_k's for now):

$$\Theta^* = \arg\min_{\Theta} F(\Theta; \mathcal{D}) + \lambda \sum_k \|\theta_k\|_2^2, \quad (3)$$

where λ is a hyperparameter to control the impact of weight decay. The weight decay term in (3) penalizes more heavily on large weights, preventing them from growing too large (Fig. 2c) [44, 55]. That said, weight decay encourages learning small balanced weights, as demonstrated by Fig. 2. Somewhat surprisingly, weight decay is underexplored in the literature of long-tailed recognition. To the best of our knowledge, existing methods did not properly tune weight decay [19, 71] (cf. code [11, 18, 70]) aside from their technical innovations. This makes it unclear whether their improved LTR performance is due to better regularization inherent in these methods. Importantly, our exploration demonstrates that, by simply tuning weight decay, we outperform most of the state-of-the-art methods on long-tailed benchmarks (Tables 2 and 3)!

MaxNorm Constraint. The third regularizer we explore is the MaxNorm constraint [25, 35, 66]. MaxNorm caps weight norms within an L2-norm ball with radius δ:

$$\Theta^* = \arg\min_{\Theta} F(\Theta; \mathcal{D}), \quad s.t. \quad \|\theta_k\|_2^2 \leq \delta^2, \ \forall\, k, \quad (4)$$

where the hyperparameter δ is the radius of the norm-ball. Solving (4) can be efficiently done through Projected Gradient Descent (PGD), which projects big weights that are outside the L2-norm ball onto the constraint set [66]. It simply applies a renormalization step after each batch update. Specifically, at each iteration, PGD first computes an updated θ_k and then projects it onto the norm ball:

$$\theta_k \leftarrow \min\left(1, \delta/\|\theta_k\|_2\right) * \theta_k. \quad (5)$$
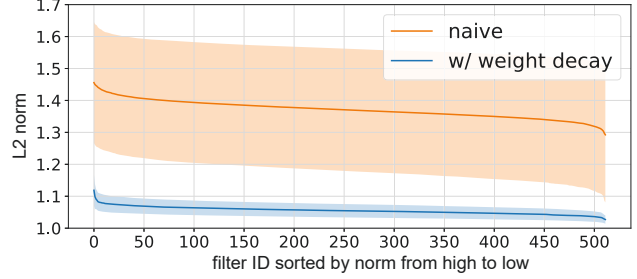

Figure 3. **Weight decay helps learn balanced weights at hidden layers**. We compare the norm distribution at each layer (which has 512 filters) from the naive model (orange) and the one trained with weight decay (blue). For each layer of a model, we sort the filter weights of each layer from high to low, compute their mean (the centerline) and variance (the shadow). While individual filters in the hidden layers are not class-specific by design, recent work demonstrates that certain filters tend to fire on certain classes [2, 81]; we still find them to be "imbalanced" in norms from the naive model. Weight decay encourages learning small and balanced filters, cf. its flat centerline and small variance.

Different from L2-normalization that strictly sets the norm value for all the filter weights as 1, MaxNorm relaxes this constraint that allows the weights to move within the norm-ball during training, as visualized in Fig. 2d.

3.2. Further Discussion

To better understand how and why the aforementioned regularizers work for long-tailed recognition, we discuss the following aspects.

Weight Decay and MaxNorm. Both regularizers balance weight norms *dynamically* during training, as opposed to L2-normalization which simply forces per-filter weights to be unit-length in norm. Weight decay encourages learning small weights, and MaxNorm encourages weights to grow within a norm ball but cap them when their norms exceed the radius. Weight decay pulls all weights to the origin. As a result, when λ increases in (3), the weight decay penalty prevails $F(\Theta; \mathcal{D})$, making training unstable [6] (Fig. 4). In contrast, MaxNorm does not pull weights towards the origin but simply caps the weight norms, and so has better numerical stability.

Although weight decay and MaxNorm appear to be quite different, they are related that weight decay can be thought of as an immediate step when solving MaxNorm. Let's rewrite the MaxNorm constrained objective function (4) by constructing a Lagrangian function:

$$\Theta^* = \arg\min_{\Theta} \max_{\gamma \geq 0} F(\Theta; \mathcal{D}) + \sum_k \gamma(\|\theta_k\|_2^2 - \delta), \quad (6)$$

where γ is the Karush–Kuhn–Tucker (KKT) multiplier. Suppose that we could solve (6) using the coordinate descent method, i.e., iteratively optimizing over Θ and γ [75]. When fixing γ, we have the same loss as (3) which is
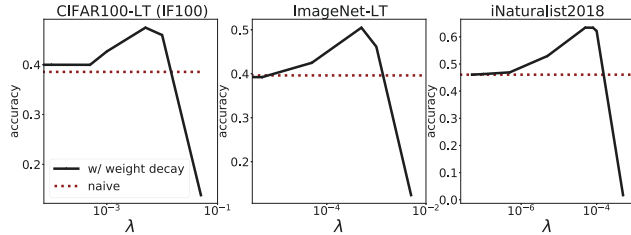
Figure 4. Tuning weight decay drastically improves long-tailed recognition performance. We do not use any class-balancing techniques but simply use CE loss and tune weight decay λ to regularize all network weight. For example, tuning λ yields 46.1% accuracy on CIFAR100-LT (IF=100), outperforming many state-of-the-art methods such as DiVE (45.4%) [32] and SSD (46.0%) [46]. By checking the publicly available code, we find that existing methods do not tune weight decay, e.g., [12, 19] set λ =2e-4 (according to their code [11, 18]), leading to poor accuracy 38.32%.

constrained by weight decay, and γ becomes the hyperparameter λ to control weight decay. That said, solving the weight decay constrained problem (3) is a step of solving MaxNorm (4). Interestingly, we find that applying weight decay and MaxNorm jointly yields better performance than using each of them independently. This is probably because of their complementary advantages: (1) weight decay on the small weights still improves their generalization and reduces overfitting, and (2) MaxNorm prevents the large weights from dominating the training.

Extreme cases. When $\delta \to \infty$ in MaxNorm, (4) boils down to the naive training (1). On the other hand, a sufficiently small δ encourages all the weights to be close to the surface of the norm-ball. This is still different from the L2-normalization which strictly requires the weights to be on the surface. Compared to L2-normalization (Fig. 2b), MaxNorm offers freespace within the norm ball to let weights grow (Fig. 2d). This intuitively explains why MaxNorm performs better than L2-normalization.

Weight decay can easily balance all network weights. We point out that weight decay regularizes classifier weights without the need to separate per-class filters. This offers convenience in training, differently from MaxNorm which must separate each filter and scaling it w.r.t its norms. Because of such a convenience, weight decay can be easily used to balance all network weights (Fig. 3). In principle, MaxNorm can also be applied to all layers, but we find it non-trivial to do so, as this seems to require setting per-layer thresholds in (4) (tuning which is time-consuming). While weight decay is widely used in network training, we find that properly tuning it drastically improves long-tailed recognition accuracy (Table 1).

3.3. Training Pipeline

Because the aforementioned weight balancing techniques are not exclusive to each other, in principle, one
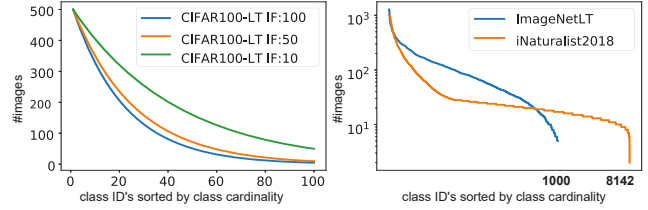

Figure 5. Frequency distributions w.r.t class cardinality of five benchmarks. **Left:** We modify CIFAR100 by downsampling examples per class with different imbalance factors (IF) varying from 10 to 100. **Right:** We use two large-scale datasets: ImageNet-LT [49] that downsamples per-class images from ImageNet [20], and iNaturalist [72] which is a real-world dataset with IF=500.

can use a single technique or multiple ones together. Recall that we follow the two-stage training paradigm [38] in our work, which first trains a network for feature representation and then trains the classifier atop the learned features. This raises a question how to apply the weight balancing techniques effectively. Among extensive exploration, we find that tuning λ for weight decay in (3) is sufficient to learn a generalizable feature representation as the first-stage training. In contrast, applying MaxNorm is nontrivial because we find that it requires setting per-layer thresholds in (4). This tuning process is time-consuming. In the second-stage training (i.e., training the classifier), we find that tuning either/both weight decay and MaxNorm remarkably improves LTR accuracy. Because the classifier training simply involves only one layer (or two layers if we think of the top two as a non-linear classifier), tuning hyperparameters of the regularizers is quite efficient. To tune them, one can use random search [5] or Bayesian Optimization [58, 69]. We use the latter in this work. In summary, our simple training pipeline consists of the following two stages:

1. *Feature learning*: train a network by using the cross-entropy loss and tuning weight decay.
2. *Classifier learning*: train a classifier over the learned features using a class-balanced loss [19], weight decay, and MaxNorm.

4. Experiments

We carry out extensive experiments to demonstrate how balancing network weights boosts long-tailed recognition performance. First, we ablate the design choices in our pipeline as suggested in Section 3.3. Then, we benchmark our methods on five established long-tailed datasets, showing that they rival or outperform existing LTR methods. We start with the experiment setup.

4.1. Experiment Setup

Datasets. We use five long-tailed benchmarks. Following [13], we modify the CIFAR100 dataset [42] by downsampling per-class training examples using some ex-
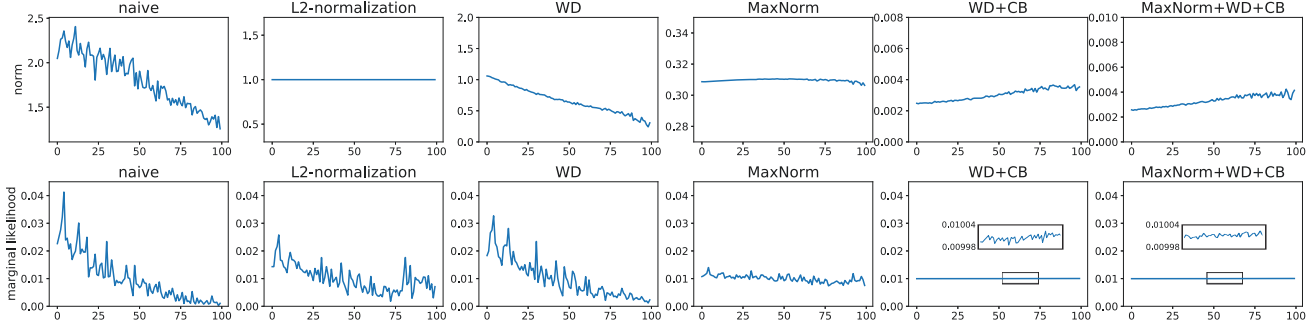
Figure 6. Per-class weight norms (**top row**) and marginal likelihood (**bottom row**) in the classifier vs. class ID sorted by class cardinality in decreasing order. The plots are on CIFAR100-LT (IF100) val-set which has class-balanced data. According to [61], the ideal marginal likelihood should follow a uniform distribution. Interestingly, L2-normalization that "perfectly" balances weight norms does not produce "uniform" marginal likelihood. Weight decay (WD) slightly mitigates norm imbalance and marginal likelihood imbalance, but MaxNorm dramatically helps both. The final model that incorporates MaxNorm, weight decay, and class-balanced loss yields nearly "perfect" marginal likelihood and balanced weights, which have a small bias towards rare-classes, presumably to emphasize their accuracy.

ponential decay functions, resulting in a long-tailed version, named *CIFAR100-LT*. CIFAR100-LT still has 100 classes and a balanced validation set for evaluation. By varying an imbalance factor (IF) $\in [100, 50, 10]$, we create three long-tailed training sets (Fig. 5-left). *ImageNet-LT* is introduced in [48] by artificially truncating the balanced version ImageNet [20]. ImageNet-LT has 1,000 classes, and the number of per-class training data ranges from 5 to 1280. *iNaturalist2018* [72] is a real-world dataset that has 8,142 naturally long-tailed classes. Fig. 5 summarizes the class frequency distributions of these datasets. ImageNet and iNaturalist2018 are publicly available for non-commercial research and educational purposes; CIFAR100 is released under the MIT license. We note that ImageNet and CIFAR100 have a "people" class or contain images that captured human faces and person signatures. This is a concern related to fairness and privacy. Therefore, we cautiously proceed our research and release our code under the MIT License without re-distributing the data.

Network architectures. For a fair comparison to prior art, we follow [19, 37, 38, 49, 78] to use specific network architectures on each dataset. We use ResNet32 [31] on CIFAR100-LT, ResNeXt50 [77] on ImageNet-LT, and ResNet50 [31] on iNaturalist2018.

Evaluation protocol. On each dataset, we train on the long-tailed class-imbalanced training set and evaluate on its (balanced) validation/test set. On ImageNet-LT, we tune hyperparameters and select models on its val-set and report performance on the test-set. On CIFAR100-LT and iNaturalist, which only have train-val sets, we follow the literature [49] that uses the val-sets to tune and benchmark. Following [49], we further report accuracy on three splits of classes that have varied numbers of training data: *Many* (>100), *Medium* ($20\sim100$), and *Few* (<20).

Implementation. We train our models using PyTorch toolbox [59] on GeForce GTX 2080Ti GPUs. The total time

spent on this work is $\sim$2 GPU years with respect to this GPU type. We train each model for 200 epochs, with batch size as 64 (for CIFAR and ImageNet-LT) / 512 (for iNaturalist), SGD optimizer with momentum 0.9, and cosine learning rate scheduler [51] that gradually decays learning rates from 0.01 to 0. We also use random left-right flipping and cropping as our training augmentation.

4.2. Ablation Study

We study (1) the impact of weight decay in LTR, (2) how to regularize classifier learning, (3) classifier weight norms and marginal likelihood distribution, and (4) the evolution of weight norms during training. We use CIFAR100-LT (IF=100) for this study (unless stated otherwise).

Weight decay. We set a single constant λ for all network parameters and focus on the first-stage training only, i.e., we use the standard cross-entropy loss to train a single network for classification. Fig. 4 draws the top-1 accuracy as a function of λ on the validation sets of three benchmarks. Clearly, tuning λ boosts accuracy, even outperforming many state-of-the-art methods (cf. Tables 2 and 3)! Moreover, the optimal λ varies for different datasets – larger datasets need a smaller weight decay, intuitively because learning over more data helps generalization and so needs less regularization.

How to regularize classifier learning. To study how to apply the balancing techniques in the second-stage learning for classifiers, we also include τ-normalization [38] because it is an effective non-learned technique that post-hoc scales the classifier learned in the first stage. We present salient conclusions based on the results in Fig. 6 (more in the supplement). First, with an improved backbone (owing to a properly tuned weight decay in the first stage), τ-norm boosts from 42.00% to 51.31%! This demonstrates the importance of learning a backbone that has balanced weights (Fig. 3). Second, it is crucial to use a class-balanced (CB)

loss [19] to learn the classifier. However, solely using the CB loss without regularizers only slightly improves (from 46.08% to 47.09%); once regularized with weight decay, it boosts to 52.42%. Third, applying both MaxNorm and weight decay improves further (53.35%), and learning more layers (as a non-linear MLP classifier) improves to 53.55%.

Classifier's weight norms and marginal likelihood. Inspired by [61], we examine the marginal likelihood based on predictions on the (balanced) test-set, on which the ideal marginal likelihood follows a uniform distribution [61]. We plot the marginal likelihood in Fig. 6, alongside the norm distribution of different models. Interestingly, L2-normalization that "perfectly" balances classifier weights does not produce balanced marginal likelihood. In contrast, MaxNorm significantly helps learn balanced weights and balanced marginal likelihood. Combining MaxNorm, weight decay, and the CB loss, the model makes nearly "perfect" marginal likelihood with a small bias toward rare-classes in weight norms, presumably because it learns to emphasize rare-class accuracy.

Weight norm evolution during training. Fig. 2 depicts how classifier's weight norms evolve during training for different models. Briefly, without regularization, weights in the naive model grow fast in norm. In contrast, weight decay prevents weights from growing too large, and MaxNorm quickly caps weights on a norm-ball surface and allows small weights to grow within the ball.

4.3. Benchmark Results

Compared Methods. Considering the rapid evolution of the LTR field [84], we compare against most relevant methods. We choose methods that are recently published and representative of different types, such as Focal [47] for loss reweighting, PaCo [17] for self-supervised pretraining and aggressive data augmentation, RIDE [73] for ensembling expert models, SSD [46] and DiVE [32] for transfer learning, etc. For comparison, we report our methods including the naive model, the one trained with properly tuned weight decay, and models that have the second-stage learning for classifier with regularizers. Tables 2 and 3 list benchmarking results on the CIFAR100-LT datasets, and ImageNet-LT and iNaturalist, respectively.

Results. Without bells and whistles, simply tuning weight decay (WD) in the first-stage training significantly boosts LTR performance over naive training and outperforms many prior methods. For example, on CIFAR100-LT (IF100) in Table 2, our WD model achieves 46.08%, outperforming the naive model (38.38%) and most of the compared methods including SSD (46.00%) [46] and DiVE (45.35%) [32]. With the second stage (classifier learning), simply post-hoc modifying (without learning) the classifier (learned in the first stage) significantly improves performance from 46.09% to 49.60% (by L2-normalization)

Table 1. **Ablation study** on CIFAR100-LT (IF=100) w.r.t top-1 accuracy (%). "CE": cross-entropy loss; "CB": class-balanced loss [19]; "WD": weight decay; "Max": MaxNorm constraint; "τ-norm": τ-normalization [38]; "+": fine-tuning the last layer(s) as the second-stage training. Here are salient conclusions. (1) Learning with a properly tuned WD boosts performance from 38.38% to 46.08%, which is +8% increase. (2) Re-training the last layer with CB and WD gives another boost (+6%) to 52.42%. (3) Based on the above, applying additional MaxNorm yields a slight improvement +1% (53.35%); finetuning the last two layers achieves 53.55%. (4) Finetuning more layers performs worse (cf. the supplement), presumably because CB induces modified gradients that affect feature learning and so hurt the final LTR performance.

Model	Many	Medium	Few	All
on the last layer (classifier)				
WD=0 (w/ CE)	64.05	35.80	11.43	38.38
+ τ-norm	59.54	38.23	25.93	42.00
WD tuned (w/ CE)	76.94	44.28	12.17	46.08
+ τ-norm	73.11	47.69	30.10	51.31
+ L2norm	76.09	47.74	20.87	49.60
+ CE & L2norm	76.37	48.11	21.00	49.87
+ CE & WD	76.97	45.94	14.00	47.22
+ CB	**77.00**	45.89	13.60	47.09
+ CB & L2norm	76.43	48.20	21.60	50.10
+ CB & WD	72.77	49.74	31.80	52.42
+ CB & Max	76.49	49.23	20.67	50.20
+ CB & WD & Max	72.60	**51.86**	32.63	53.35
on the last two layers				
+ CB & WD & Max	71.37	51.17	**35.53**	**53.55**

and to 51.31% (by τ-normalization). By learning the classifier regularized with MaxNorm and/or weight decay, we achieve the state of the art (53.35%). Such a conclusion holds on all benchmarks. However, on the two large-scale datasets ImageNet-LT and iNaturalists in Table 3, our methods rival prior art but underperform two types of methods that have "bells and whistles", including ensemble methods (RIDE [73] and ACE [46]) that learn and fuse multiple models, and self-supervised learning based methods (PaCo [17] and SSD [46]) that adopt aggressive data augmentation techniques [16, 30].

5. Conclusion

Long-tailed recognition (LTR) is a crucial challenge for real-world data that tends to be imbalanced. Our work is motivated by the empirical observation that a model naively trained over long-tailed data has artificially large weights for common classes (because they have more data to train than rare classes). We propose to learn balanced weights via parameter regularization, including weight decay and MaxNorm regularizers. Our extensive study shows that properly applying these regularizers greatly boosts LTR performance. We introduce a simple approach that outperforms prior art on five long-tailed benchmarks. Because these reg-

Table 2. **Benchmarking on CIFAR-100-LT** with different imbalance factors $[100, 50, 10]$ w.r.t top-1 accuracy (%). Please refer to the caption of Table 1 for abbreviations; "+" methods use CB loss. WD makes a substantial impact on the training of LTR networks. Finetuning the classifier with proper regularization improves much further. This clearly shows the significance of parameter regularization in balancing weights for LTR. Somewhat surprisingly, properly tuning weight decay in the two-stage training paradigm outperforms all existing methods on these three datasets.

imbalance factor	100	50	10
CE [19]	38.32	43.85	55.71
CE+CB [19]	39.60	45.32	57.99
KD [33]	40.36	45.49	59.22
LDAM-DRW [12]	42.04	46.62	58.71
BBN [88]	42.56	47.02	59.12
LogitAjust [54]	42.01	47.03	57.74
LDAM+SSP [78]	43.43	47.11	58.91
Focal [47]	38.41	44.32	55.78
Focal+CB [19]	39.60	45.17	57.99
De-confound [71]	44.10	50.30	59.60
τ-norm [38]	47.73	52.53	63.80
SSD [46]	46.00	50.50	62.30
DiVE [32]	45.35	51.13	62.00
DRO-LT [65]	47.31	57.57	63.41
PaCo [17]	52.00	56.00	64.20
ACE (4-expert) [10]	49.60	51.90	—
RIDE (4-expert) [73]	49.10	—	—
Our methods (weight balancing)			
naive	38.38	43.99	57.31
WD	46.08	52.71	66.03
+ L2norm	49.60	56.33	67.16
+ τ-norm	51.31	57.65	67.79
+ WD	52.42	57.47	67.96
+ Max	50.24	56.06	67.10
+ WD & Max	**53.35**	**57.71**	**68.67**

Table 3. **Benchmarking on ImageNet-LT and iNaturalists** in top-1 accuracy (%). Please refer to Table 2 for methods' names and salient conclusions. We list the numbers of compared methods reported in their respective papers. Overall, our simple approach achieves competitive results with the prior methods particularly when they train a "single (expert)" model, although underperforms a few recent state-of-the-art methods which train and ensemble expert models (RIDE [73] and ACE [10]), or adopt self-supervised pretraining (e.g., PaCo [17] and SSD [46]) with aggressive data augmentation techniques [16, 30].

	ImageNet-LT				iNaturalist			
	Many	Med.	Few	All	Many	Med.	Few	All
CE [38]	65.9	37.5	7.7	44.4	72.2	63.0	57.2	61.7
CE+CB [19]	39.6	32.7	16.8	33.2	53.4	54.8	53.2	54.0
KD [33]	58.8	26.6	3.4	35.8	72.6	63.8	57.4	62.2
Focal [19]	36.4	29.9	16.0	30.5	—	—	—	61.1
OLTR [49]	43.2	35.1	18.5	35.6	59.0	64.1	64.9	63.9
LFME [76]	47.1	35.0	17.5	37.2	—	—	—	—
BBN [88]	—	—	—	—	49.4	70.8	65.3	66.3
cRT [38]	61.8	46.2	27.3	49.6	69.0	66.0	63.2	65.2
τ-norm [38]	59.1	46.9	30.7	49.4	65.6	65.3	65.5	65.6
De-confound [71]	62.7	48.8	31.6	51.8	—	—	—	—
DiVE [32]	64.1	50.4	31.5	53.1	70.6	70.0	67.6	69.1
DRO-LT [65]	64.0	49.8	33.1	53.5	—	—	—	69.7
DisAlign [83]	61.3	52.2	31.4	52.9	69.0	71.1	70.2	70.6
Our methods (weight balancing)								
naive	55.3	31.4	12.5	38.0	54.7	46.0	43.9	46.1
WD	68.5	42.4	14.2	48.6	74.5	66.5	61.5	65.4
+ L2norm	61.2	48.9	42.6	52.8	11.2	47.4	66.9	51.3
+ τ-norm	64.0	49.0	36.3	53.1	71.3	69.8	68.9	69.6
+ WD	62.0	49.7	41.0	53.3	71.0	70.3	69.4	70.0
+ Max	62.2	50.1	37.5	53.0	71.4	68.9	69.1	69.2
+ WD & Max	62.5	50.4	41.5	53.9	71.2	70.4	69.7	70.2
SOTA with "bells and whistles": ensembles, data augmentation, and self-supervised pretraining								
RIDE [73]	67.9	52.3	36.0	56.1	66.5	72.1	71.5	71.3
ACE [10]	—	—	—	56.6	—	—	—	72.9
SSD [46]	66.8	53.1	35.4	56.0	—	—	—	71.5
PaCo [17]	63.2	51.6	39.2	54.4	69.5	72.3	73.1	72.3

ularizers are underexplored in the long-tailed literature, we hope our study draws attention from the practitioners that parameter regularization should be the first method to consider, when addressing real-world problems related to the long-tailed distribution.

Limitations. While we focus on the orthogonal direction of parameter regularization to address LTR, we have not studied how our approaches complement existing techniques. For example, how to balance weights in training each of the expert models, or how to balance the weights alongside sophisticated data augmentation and self-supervised pretraining. We also point out that other regularization techniques might be better at balancing weights, for example using Lp-norm weight decay where $p \neq 2$ [3]. We leave them to future work.

Societal Impact. Because real-world data tends to follow long-tailed distributions, our work has multiple positive societal impacts. For example, addressing the long-tail proves an important direction for studying bias and fairness in recognition [15]. However, any system that makes it easier to train a fair classifier on long-tailed classes also makes it possible for a malicious agent to train a system that automatically discriminates against a certain subgroup for which only little training data is available. This is potentially a negative societal impact.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2019. 3

[2] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR*, 2017. 3, 4

[3] Agnes Benedek and Rafael Panzone. The space lp, with mixed norm. *Duke Mathematical Journal*, 28(3):301–324, 1961. 8

[4] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *NeurIPS*, 2007. 3

[5] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(2), 2012. 5

[6] Dimitri P Bertsekas. Multiplier methods: A survey. *Automatica*, 12(2):133–145, 1976. 4

[7] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. 3

[8] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018. 2

[9] Peter Bühlmann and Sara Van De Geer. *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011. 3

[10] Jiarui Cai, Yizhou Wang, and Jenq-Neng Hwang. Ace: Ally complementary experts for solving long-tailed recognition in one-shot. In *ICCV*, 2021. 1, 2, 8

[11] Kaidi Cao. https : / / github . com / kaidic / LDAM - DRW / blob / 3193f05c1e6e8c4798c5419e97c5a479d991e3e9/ cifar_train.py#L54. *commit 6feb304*, 2019. 4, 5

[12] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, 2019. 1, 2, 5, 8

[13] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. Deep quantization network for efficient image retrieval. In *AAAI*, 2016. 5

[14] 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. 2

[15] Irene Chen, Fredrik D Johansson, and David Sontag. Why is my classifier discriminatory? In *NeurIPS*, 2018. 8

[16] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *NeurIPS*, 2020. 7, 8

[17] Jiequan Cui, Zhisheng Zhong, Shu Liu, Bei Yu, and Jiaya Jia. Parametric contrastive learning. In *ICCV*, 2021. 1, 2, 7, 8

[18] Yin Cui. https : / / github . com / richardaecn / class - balanced - loss / blob / 1d7857208a2abc03d84e35a9d5383af8225d4b4d/

[19] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019. 2, 3, 4, 5, 6, 7, 8

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5, 6

[21] 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, pages 1–8. Citeseer, 2003. 2

[22] Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36, 2004. 2

[23] Chengjian Feng, Yujie Zhong, and Weilin Huang. Exploring classification equilibrium in long-tailed object detection. In *ICCV*, 2021. 2

[24] Ian Goodfellow, Y Bengio, and A Courville. Regularization for deep learning. *Deep learning*, pages 216–261, 2016. 3

[25] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016. 3, 4

[26] Hao Guo and Song Wang. Long-tailed multi-label visual recognition by collaborative training on uniform and re-balanced samplings. In *CVPR*, 2021. 2

[27] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 2

[28] 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*, pages 878–887. Springer, 2005. 2

[29] Stephen Hanson and Lorien Pratt. Comparing biases for minimal network construction with back-propagation. *NeurIPS*, 1988. 2, 3

[30] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 7, 8

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3, 6

[32] Yin-Yin He, Jianxin Wu, and Xiu-Shen Wei. Distilling virtual examples for long-tailed recognition. In *ICCV*, 2021. 5, 7, 8

[33] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015. 8

[34] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 3

[35] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012. 2, 3, 4

[36] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Deep imbalanced learning for face recognition and attribute prediction. *PAMI*, 42(11):2781–2794, 2019. 2

src/cifar_main.py#L581. *commit 0ab6eb7*, 2019. 4, 5

[37] 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 *CVPR*, 2020. 2, 6

[38] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *ICLR*, 2020. 2, 3, 4, 5, 6, 7, 8

[39] Salman Khan, Munawar Hayat, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Striking the right balance with uncertainty. In *CVPR*, 2019. 2

[40] 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. 2

[41] Shu Kong and Deva Ramanan. Opengan: Open-set recognition via open data generation. In *ICCV*, 2021. 2

[42] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012. 3

[44] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *NeurIPS*, 1992. 2, 3, 4

[45] Jan Kukavcka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *arXiv:1710.10686*, 2017. 3

[46] Tianhao Li, Limin Wang, and Gangshan Wu. Self supervision to distillation for long-tailed visual recognition. In *ICCV*, 2021. 5, 7, 8

[47] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 2, 7, 8

[48] Si Liu, Risheek Garrepalli, Thomas G Dietterich, Alan Fern, and Dan Hendrycks. Open category detection with pac guarantees. In *ICML*, 2018. 2, 6

[49] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, 2019. 2, 5, 6, 8

[50] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *ICLR*, 2017. 3

[51] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 6

[52] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 3

[53] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018. 2

[54] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *ICLR*, 2021. 8

[55] John E Moody. Note on generalization, regularization and architecture selection in nonlinear learning systems. In *Neural Networks for Signal Processing Proceedings of IEEE Workshop*. IEEE, 1991. 4

[56] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012. 3

[57] Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *ICML*, 2004. 3

[58] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014. 5

[59] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[60] William J Reed. The pareto, zipf and other power laws. *Economics letters*, 74(1):15–19, 2001. 2

[61] Jiawei Ren, Cunjun Yu, Shunan Sheng, Xiao Ma, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Balanced meta-softmax for long-tailed visual recognition. In *NeurIPS*, 2020. 3, 6, 7

[62] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018. 2

[63] Ingrid C Romero, Shu Kong, Charless C Fowlkes, Carlos Jaramillo, Michael A Urban, Francisca Oboh-Ikuenobe, Carlos D'Apolito, and Surangi W Punyasena. Improving the taxonomy of fossil pollen using convolutional neural networks and superresolution microscopy. *Proceedings of the National Academy of Sciences*, 117(45):28496–28505, 2020. 2

[64] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*, 2016. 3

[65] Dvir Samuel and Gal Chechik. Distributional robustness loss for long-tail learning. In *ICCV*, 2021. 8

[66] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011. 2, 3, 4

[67] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016. 2

[68] 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 *NeurIPS*, 2019. 2

[69] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *NeurIPS*, 2012. 5

[70] Kaihua Tang. https://github.com/KaihuaTang/Long-Tailed-Recognition.pytorch/blob/90c8b2c0b66d17f78b67263861bc9d858fe20128/classification/config/CIFAR100_LT/feat_unifrom.yaml#L22. *commit 54c07cf*, 2020. 4

[71] Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. Long-tailed classification by keeping the good and removing the bad momentum causal effect. In *NeurIPS*, 2020. 2, 3, 4, 8

[72] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018. 2, 5, 6

[73] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella X Yu. Long-tailed recognition by routing diverse distribution-aware experts. In *ICLR*, 2020. 1, 2, 7, 8

[74] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *NeurIPS*, 2017. 2

[75] Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015. 4

[76] Liuyu Xiang, Guiguang Ding, and Jungong Han. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *ECCV*, 2020. 2, 8

[77] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 6

[78] Yuzhe Yang and Zhi Xu. Rethinking the value of labels for improving class-imbalanced learning. In *NeurIPS*, 2020. 2, 6, 8

[79] Xi Yin, Xiang Yu, Kihyuk Sohn, Xiaoming Liu, and Manmohan Chandraker. Feature transfer learning for face recognition with under-represented data. In *CVPR*, 2019. 2

[80] Zhuoning Yuan, Yan Yan, Rong Jin, and Tianbao Yang. Stagewise training accelerates convergence of testing error over sgd. *arXiv:1812.03934*, 2018. 3

[81] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 3, 4

[82] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 3

[83] Songyang Zhang, Zeming Li, Shipeng Yan, Xuming He, and Jian Sun. Distribution alignment: A unified framework for long-tail visual recognition. In *CVPR*, 2021. 2, 8

[84] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *arXiv:2110.04596*, 2021. 2, 7

[85] Yunhan Zhao, Shu Kong, and Charless Fowlkes. Camera pose matters: Improving depth prediction by mitigating pose distribution bias. In *CVPR*, 2021. 2

[86] Yunhan Zhao, Shu Kong, Daeyun Shin, and Charless Fowlkes. Domain decluttering: Simplifying images to mitigate synthetic-real domain shift and improve depth estimation. In *CVPR*, 2020. 3

[87] Yaoyao Zhong, Weihong Deng, Mei Wang, Jiani Hu, Jianteng Peng, Xunqiang Tao, and Yaohai Huang. Unequaltraining for deep face recognition with long-tailed noisy data. In *CVPR*, 2019. 2

[88] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *CVPR*, 2020. 2, 8