AutoBalance: Optimized Loss Functions for Imbalanced Data

Mingchen Li Xuechen Zhang
University of California, Riverside
{mli176,xzhan394}@ucr.edu

Christos Thrampoulidis
University of British Columbia
cthrampo@ece.ubc.edu.ca

Jiasi Chen
University of California, Riverside
jiasi@cs.ucr.edu

Samet Oymak
University of California, Riverside
oymak@ece.ucr.edu

Abstract

Imbalanced datasets are commonplace in modern machine learning problems. The presence of under-represented classes or groups with sensitive attributes results in concerns about generalization and fairness. Such concerns are further exacerbated by the fact that large capacity deep nets can perfectly fit the training data and appear to achieve perfect accuracy and fairness during training, but perform poorly during test. To address these challenges, we propose AutoBalance, a bi-level optimization framework that automatically designs a training loss function to optimize a blend of accuracy and fairness-seeking objectives. Specifically, a lower-level problem trains the model weights, and an upper-level problem tunes the loss function by monitoring and optimizing the desired objective over the validation data. Our loss design enables personalized treatment for classes/groups by employing a parametric cross-entropy loss and individualized data augmentation schemes. We evaluate the benefits and performance of our approach for the application scenarios of imbalanced and group-sensitive classification. Extensive empirical evaluations demonstrate the benefits of AutoBalance over state-of-the-art approaches. Our experimental findings are complemented with theoretical insights on loss function design and the benefits of train-validation split. All code is available open-source.

1 Introduction

Recently, deep learning, large datasets, and the evolution of computing power have led to unprecedented success in computer vision, and natural language processing [15, 39, 62]. This success is partially driven by the availability of high-quality datasets, built by carefully collecting a sufficient number of samples for each class. In practice, real-world datasets are frequently imbalanced and exhibit long-tailed behavior, necessitating a careful treatment of the minorities [20, 53, 23]. Indeed, modern classification tasks can involve thousands of classes, so it is perhaps intuitive that some classes should be over/under-represented compared to others. Besides class imbalance, minorities can also appear at the feature-level; for instance, the specific values of the features of an example can vary depending on that example’s membership in certain sensitive or protected groups, e.g. race, gender, disabilities (see also Figure 1a). In scenarios where imbalances are induced by heterogeneous client datasets (e.g., in the context of federated learning), addressing these imbalances can help ensure that a machine learning model works well for all clients, rather than just those that generate the majority of the training data. This rich set of applications motivate the careful treatment of imbalanced datasets.

In the imbalanced classification literature, the recurring theme is maximizing a fairness-seeking objective, such as balanced accuracy. Unlike standard accuracy, which can be dominated by the majorities, a fairness-seeking objective seeks to promote examples from minorities, and downweigh
How to design a training loss to maximize a fairness-seeking objective on the test set?

A classical answer to this question is to use a Bayes-consistent loss functions. For instance, weighted cross-entropy (e.g., each class gets a different weight, see Sec. 2) is traditionally a good choice for optimizing weighted accuracy objectives. Unfortunately, this intuition starts to break down when the training problem is overparameterized, which is a common practice in deep learning: in essence, for large capacity deep nets, the training process can perfectly fit to the data, and training loss is no longer indicative of test error. In fact, recent works [8, 38] show that weighted cross-entropy has minimal benefit to balanced accuracy, and instead alternative methods based on margin adjustment can be effective (namely, by ensuring that minority classes are further away from decision boundary). These ideas led to the development of a parametric cross-entropy function \( \ell_k(x, y) = w_k \log(1 + \sum_{k \neq y} e^{l_k - l_y} e^{\Delta_k f_k(x)} - \Delta_k f_y(x)) \), which allows for a personalized treatment of the individual classes via the design parameters \((w_k, l_k, \Delta_k)^K\) \([8, 38, 35, 67]\). Here, \(w_k\) is the classical weighting term whereas \(l_k\) and \(\Delta_k\) are additive and multiplicative logit adjustments. However, despite these developments, it is unclear how such parametric cross-entropy functions can be tuned for use for different fairness objectives, for example to tackle class or group imbalances. The works by [8, 53] provide theoretically-motivated choices for \((w_k, l_k)\), while [38] argues that \((w_k, l_k)\) is not as effective as \(\Delta_k\) in the interpolating regime of zero training error and proposes the simultaneous use of all three different parameter types. However, these works do not provide an optimized loss function that can be systematically tailored for different fairness objectives, such as balanced accuracy common in class imbalanced scenarios, or equal opportunity [23, 17] which is relevant in group-sensitive settings.

In this work, we address these shortcomings by designing the loss function within the optimization in a principled fashion, to handle different fairness-seeking objectives. Our main idea is to use bi-level optimization, where the model weights are optimized over the training data, and the loss function is automatically tuned by monitoring the validation loss. Our core intuition is that unlike training data, the validation data is difficult to fit and will provide a consistent estimator of the test objective. Therefore, the fundamental question of interest is:

\[ \text{How to design a training loss to maximize a fairness-seeking objective on the test set?} \]

**Contributions.** Based on this high-level idea, this paper takes a step towards a systematic treatment of imbalanced learning problems with contributions along several fronts: state-of-the-art performance, data augmentation, applications to different imbalance types, and theoretical intuitions. Specifically:
- We introduce **AutoBalance**—a bilevel optimization framework— that designs a fairness-seeking loss function by jointly training the model and the loss function hyperparameters in a systematic way (Figure 1b, Section 2). We introduce novel strategies that narrow down the search space to improve convergence and avoid overfitting. To further improve the performance, our design also incorporates data augmentation policies personalized to subpopulations (classes or groups). We demonstrate the benefits of AutoBalance when optimizing various fairness-seeking objectives over the state-of-the-art, such as logit-adjustment (LA) [53] and label-distribution-aware margin (LDAM) [8] losses. The code is available online [42].
Algorithm 1: AutoBalance via Bilevel Optimization

**Input:** Model \( f_\theta \) with weights \( \theta \), dataset \( S = S_T \cup S_V \), step sizes \( \eta_{\alpha} \) & \( \eta_\theta \), # iterations \( t_2 > t_1 \)

1. Initialize \( \alpha \) with \( \ell_{\text{fair}}(\cdot) = \ell_{\text{train}}(\cdot; \alpha) \) // Consistent initialization
2. Train \( \theta \) for \( t_1 \) iterations (\( \alpha \) is fixed) // Search Phase: Starts with warm-up
3. for \( i \leftarrow t_1 \) to \( t_2 \) do
   4. Sample training batch \( B_T \) from \( S_T \);
5. \( B_T \leftarrow A(B_T) \) // Apply class-personalized augmentation
   6. \( \theta \leftarrow \theta - \eta_\theta \nabla_{\theta} \mathcal{L}_{\text{train}}^{B_T}(f_\theta; \alpha) \)
7. Sample validation batch \( B_V \) from \( S_V \);
8. Compute hyper-gradient \( \nabla_{\alpha} \mathcal{L}_{\text{fair}}^{B_V}(f_\theta) \) // via Approx. Implicit Differentiation
9. \( \alpha \leftarrow \alpha - \eta_{\alpha} \nabla_{\alpha} \mathcal{L}_{\text{fair}}^{B_V}(f_\theta) \) // Update loss function hyper-parameters
10. end
11. Set \( \alpha_* \leftarrow \alpha \), \( S_T \leftarrow S \), reset weights \( \theta \) // Retraining Phase: Use all data and \( \alpha \).
12. Train \( \theta \) for \( t_2 \) iterations using \( \alpha \).

**Result:** The final model \( \theta_* \leftarrow \theta \) and hyper-parameters \( \alpha_* \).

- Extensive experiments provide several takeaways (Section 3). First, AutoBalance discovers loss functions from scratch that are consistent with theory and intuition: hyperparameters of the minority classes evolve to upweight the training loss of minority to promote them. Second, the impact of individual design parameters in the loss function is revealed, with the additive adjustment \( \Delta_k \) and multiplicative adjustment \( \Delta_k \) synergistically improving the fairness objective. Third, personalized data augmentation can further improve the performance over a single generic augmentation policy.
- Beyond class imbalance, we consider applications of loss function design to the group-sensitive setting (Section 4). Our experiments show that AutoBalance consistently outperforms various baselines, leading to a more efficient Pareto-frontier of accuracy-fairness tradeoffs.

### 1.1 Problem Setup for Class Imbalance

We first focus on the label-imbalance problem. The extension to the group-imbalanced setting (approach, algorithms, evaluations) is deferred to Section 4. Let \([K]\) denote the set \( \{1, \ldots, K\} \). Suppose we have a dataset \( S = \{x_i, y_i\}_{i=1}^n \) sampled i.i.d. from a distribution \( D \) with input space \( \mathcal{X} \) and \( K \) classes. For a training example \( (x, y) \), \( x \in \mathcal{X} \) is the input feature and \( y \in [K] \) is the output label. Let \( f : \mathcal{X} \rightarrow \mathbb{R}^K \) be a model that outputs a distribution over classes and let \( \hat{y}_f(x) = \arg \max_{i \in [K]} f(x)_i \). The standard classification error is denoted by \( \mathcal{E}(f) = \mathbb{P}_D[y \neq \hat{y}_f(x)] \). For a loss function \( \ell(y, \hat{y}) \) (e.g. cross-entropy), we similarly denote

\[
\text{Population risk: } \mathcal{L}(f) = \mathbb{E}_D[\ell(y, \hat{y}_f(x))] \quad \text{and} \quad \text{Empirical risk: } \mathcal{L}^S(f) = \frac{1}{n} \sum_{i=1}^n \ell(y_i, \hat{y}_f(x_i)).
\]

**✓ Setting: Imbalanced classes.** Define the frequency of the \( k \)-th class via \( \pi_k = \mathbb{P}(y = k) \). Label/class-imbalance occurs when the class frequencies differ substantially, i.e., \( \max_{i \in [K]} \pi_i \gg \min_{i \in [K]} \pi_i \). Let us introduce

Balanced risk: \( \mathcal{L}_{\text{bal}}(f) = \frac{1}{K} \sum_{k=1}^K \mathcal{L}_k(f) \) and Class-conditional risk: \( \mathcal{L}_k(f) = \mathbb{E}_{D_k}[\ell(y, \hat{y}_f(x))] \).

Similarly, let \( \mathcal{E}_k(f) \) be the class-conditional classification error and \( \mathcal{E}_{\text{bal}}(f) := (1/K) \sum_{k=1}^K \mathcal{E}_k(f) \) be the balanced error. In this setting, rather than the standard test error \( \mathcal{E}(f) \), our goal is to minimize balanced error. At a high-level, we propose to do this by designing an imbalance-aware training loss that maximizes balanced validation accuracy.

### 2 Methods: Loss Functions, Search Space Design, and Bilevel Optimization

Our main goal in this paper is automatically designing loss functions to optimize target objectives for imbalanced learning (e.g., Settings A and B). We will employ a parametrizable family of loss functions that can be tailored to the needs of different classes or groups. Cross-entropy variations have been proposed by [44, 35, 16] to optimize balanced objectives. Our design space will utilize recent works which introduce Label-distribution-aware margin (LDAM) [8], Logit-adjustment (LA) [53], Class-dependent temperatures (CDT) [67], Vector scaling (VS) [38] losses. Specifically, we...
build on the following parametric loss function controlled by three vectors \( w, l, \Delta \in \mathbb{R}^K \):
\[
\ell(y, f(x)) = w_y \log \left( 1 + \sum_{k \neq y} e^{l_k - l_y} \cdot e^{\Delta_k f_y(x) - \Delta_y f_y(x)} \right).
\]
(2.1)

Here, \( w_y \) enables conventional weighted CE and \( l_y \), and \( \Delta_y \) are additive and multiplicative adjustments to the logits. This choice is same as the VS-loss introduced in [38], which borrows the \( \Delta \) term from [67] and \( l \) term from [8, 53]. [53] makes the observation that we can use \( l \) rather than \( w \) while ensuring Fisher consistency in balanced error. We make the following complementary observation.

**Lemma 1** Parametric loss function (2.1) is not consistent for standard or balanced errors if there are distinct multiplicative adjustments i.e. \( \Delta_i \neq \Delta_j \) for some \( i, j \in [K] \).

While consistency is a desirable property, it is intuitively more critical during the earlier phase of the training where the training risk is more indicative of the test risk. In the interpolating regime of zero-training error, [38] shows that \( w, l \) can be ineffective and multiplicative \( \Delta \)-adjustment can be more favorable. Our algorithm will be initialized with a consistent weighted-CE; however, we will allow the algorithm to automatically adapt to the interpolating regime by tuning \( l \) and \( \Delta \).

**Proposed training loss function.** For our algorithm, we will augment (2.1) with data augmentation that can be personalized to distinct classes. Let us denote the data augmentation policies by \( A = (A_y)_{y \in [K]} \), where each \( A_y \) stochastically augments an input example with label \( y \). Additionally, we clamp \( \Delta_y \) with the sigmoid function \( \sigma \) to limit its range to \((0, 1)\) to ensure non-negativity. To this end, our loss function for the lower-level optimization (over training data) is as follows:
\[
\ell_{\text{train}}(y, x, f; \alpha) = -\mathbb{E}_A \left[ w_y \log \left( \frac{e^{\sigma(\Delta_y) f_y(A_y(x)) + l_y}}{\sum_{i \in [K]} e^{\sigma(\Delta_i) f_i(A_i(x)) + l_i}} \right) \right].
\]
(2.2)

Here, \( \alpha \) is the set of hyperparameters of the loss function that we wish to optimize, specifically \( \alpha = [w, l, \Delta, \text{param}(A)] \), \text{param}(A) is the parameterization of the augmentation policies \( (A_y)_{y \in [K]} \).

**Personalized data augmentation (PDA).** Remarkable benefits of data augmentation techniques provide a natural motivation to investigate whether one can benefit from learning class-personalized augmentation policies. The PDA idea relates to SMOTE [9], where the minority class is over-sampled by creating synthetic examples. To formalize the benefits of PDA, consider a spherical augmentation strategy where \( A_y(x) \) samples a vector uniformly from an \( \ell_2 \)-ball of radius \( \varepsilon_y \) around \( x \). As visualized in Figure 2 for a linear classifier, if the augmentation strengths of both classes are equal, the max-margin classifier is not affected by the application of the data augmentation and remains identical. Thus, augmentation has no benefit. However by applying a stronger augmentation on minority, the decision boundary is shifted to protect minority which can provably benefit the balanced accuracy [38]. The following intuitive observation links the PDA to parametric loss (2.1).

**Lemma 2** Consider a binary classification task with labels 0 and 1 and a linearly separable training dataset. For any parametric loss (2.1) choices of \((l_1, \Delta_1, w_1)_{1=0}^1 \), there exists spherical augmentation strengths for minority-majority classes so that, without regularization, optimizing the logistic loss with personalized augmentations returns the same classifier as optimizing (2.1).

This lemma is similar in flavor to [31], which considers a larger uncertainty set around the minority class. But, as discussed in the appendix, Lemma 2 is relevant in the overparameterized regime whereas the approach of [31] is ineffective for separable data [51]. Algorithmically, the augmentations that we consider are much more flexible than the \( \ell_p \)-balls of [31] and our experiments showcase the value of our approach in state-of-the-art multiclass settings. Besides, note that the (theoretical) benefits of PDA can go well-beyond Lemma 2 by leveraging the invariances [10, 14] (via rotation, translation).

### 2.1 Proposed Bilevel Optimization Method

We formulate the loss function design as a bilevel optimization over hyperparameters \( \alpha \) and a hypothesis set \( \mathcal{F} \). We split the dataset \( \mathcal{S} \) into training \( \mathcal{S}_T \) and validation \( \mathcal{S}_V \) sets with \( n_T \) and \( n_V \)
Algorithm 1 summarizes our approach and highlights the key components. The training loss \( \Delta \), as upper and lower level optimizations, respectively.

\[ \text{Why Bilevel Optimization?} \]

We choose differentiable optimization over alternative hyperparameter tuning methods because our hyperparameter space is continuous and potentially large (e.g., in the case of ImageNet where \( K = 1,000 \)).

2.2 Reducing the Hyperparameter Search Space and the Benefits of Validation Set

Suppose we wish to optimize the hyperparameter \( \alpha = (w_y, l_y, \Delta_y)_{y=1}^{K} \) of the parametric loss (2.1). An important challenge is the dimensionality of \( \alpha \), which is proportional to the number of classes \( K \), as we need a triplet \((w_y, l_y, \Delta_y)\) for each class. For instance, ImageNet has \( K = 1,000 \).
whereas iNaturalist has \( K = 8,142 \) classes resulting in high-dimensional hyperparameters. In our experiments, we found that directly optimizing over such large spaces leads to convergence issues likely because of the difficulty of hypergradient estimation. Additionally, with large number of hyperparameters there is increased concern for validation overfitting. This is especially so for the tail classes (e.g. the smallest class in CIFAR100-LT has only 1 validation example with an 80-20% split). On the other hand, it is well-known in AutoML literature (e.g. neural architecture search \([45, \text{AutoAugment} [12]\)) that designing a good search space is critical for attaining faster convergence and good validation accuracy. To this end, we propose \textit{subspace-based search spaces} for hyperparameters \((w_i, l_i, \Delta_i)\). To explain the idea, consider the logit-adjustment parameters \( l = [l_1, \ldots l_K] \) and \( \Delta = [\Delta_1, \ldots \Delta_K] \). We propose representing these \( K \) dimensional vectors via \( K' < K \) dimensional embeddings \( l', \Delta' \) as follows

\[
\Delta = D \pi \Delta' \quad \text{and} \quad l = D \pi l' \quad \text{where} \quad l', \Delta' \in \mathbb{R}^{K'}.
\]

Here, \( D \pi \in \mathbb{R}^{K \times K'} \) is a frequency-aware dictionary that we design, and the range space of \( D \pi \) becomes the hyperparameter search space. In our algorithm, we cluster the classes in terms of their frequency and assign the same hyperparameter to classes with similar frequencies. To be concrete, if each cluster has size \( C \), then \( K' = \lceil K / C \rceil \). For CIFAR10-LT, CIFAR100-LT, ImageNet-LT and iNaturalist, we use \( C = \{1, 10, 20, 40\} \) respectively. In this scheme, each column \( d \) of the matrix \( D \pi \) is the indicator function of one of the \( K' \) clusters, i.e. \( d_i = 1 \) if \( i \)th class is within the cluster and \( 0 \) otherwise. Clusters are pictorially illustrated in Fig. 3a. Finally, we remark that the specific hyperparameter choices of CDT \([67]\), LA \([53]\), and VS \([38]\) losses in the corresponding papers, can be viewed as specific instances of the above search space design. For instance, LA-loss chooses a scalar \( w \) and sets \( l_i = \tau \log(\pi_i) \). This corresponds to a dictionary containing a single column \( d \in \mathbb{R}^{K} \) with entries \( d_i = \log(\pi_i) \).

**Why is train-validation split critical?** In Figure 4 we plot the balanced errors of training/validation/test datasets at each epoch of the search phase. The training data is fixed whereas we evaluate different validation set sizes. The first finding is that training loss always overfits (dotted) until zero error whereas validation loss mildly overfits (dashed vs solid). Second, larger validation does help improve test accuracy (compare solid lines). The training behavior is in line with the fact that large capacity networks can perfectly fit and achieve 100% training accuracy \([18, 56, 29]\). This also means that different accuracy metrics or fairness constraints can be perfectly satisfied. To truly find a model that lies on the Pareto-front of the (accuracy, fairness) tradeoff, the optimization procedure should (approximately) evaluate on the population loss. Thus, as in Figure 4, the validation phase provides this crucial test-proxy in the overparameterized setting where training error is vacuous. Following the model selection literature \([32, 33]\), the intuition is that, as the dimensionality of the hyper-parameter \( \alpha \) is typically smaller than the validation size \( n_V \), validation loss will not overfit and will be indicative of the test even if the training loss is zero. Our search space design in Section 2.2 also helps to this end by increasing the over-sampling ratio \( n_V / \dim(\alpha) \) via class clustering. In the appendix, we formalize these intuitions for multi-objective problems (e.g., accuracy + fairness). Under mild assumptions, we show that a small amount of validation data is sufficient to ensure that the Pareto-front of the validation risk uniformly approximates that of the test risk. Concretely, for two objectives \((L_1, L_2)\), uniformly over all \( \lambda \), the hyperparameter \( \alpha \) minimizing the validation risk \( L_{\text{val}}^{\alpha}(f) = (1 - \lambda) L_1^{\alpha} + \lambda L_2^{\alpha} \) in (2.3) also approximately minimizes the test risk \( L_{\text{test}}(f) \).

3 Evaluations for Imbalanced Classes

In this section, we present our experiments on various datasets (CIFAR-10, CIFAR-100, iNaturalist-2018 and ImageNet) when the classes are imbalanced. The goal is to understand whether our bilevel optimization can design effective loss functions that improve balanced error \( \epsilon_{\text{bal}} \) on the test set. The setup is as follows. \( \epsilon_{\text{bal}} \) is the test objective. The validation loss \( L_{\text{val}} \) is the balanced cross-entropy...
150 epoch training with 40 epoch warm-up before the loss function design starts. For the retraining, we change the batch size to 128 and adjust initial learning rate accordingly as suggested by [22].

Then, we reduce the search space with cluster sizes of 10, 20, and 40 as visualized in Figure 3(a) and as with standard mini-batch stochastic gradient decent (SGD) using learning rate 0.1, momentum 0.9, and weight decay 1e-4, over 300 epochs. The learning rate decays at epochs 220 and 260 with a factor 0.1. The upper-level hyper-parameter optimization computes the hyper-gradientes via implicit differentiation. Because the hyper-gradient is mostly meaningful when the network achieves near zero loss (Thm 1 of [46]), we start the validation optimization after 120 epochs of the training optimization, using SGD with initial learning rate 0.05, momentum 0.9, and weight decay 1e-4, we follow the same learning rate decay at epoch 220 and 260. For CIFAR10-LT, 20 hyper-parameters are trained, corresponding to \( l_y \) and \( \Delta_y \) of each 10 classes. For CIFAR100-LT, ImageNet-LT, and iNaturalist, we reduce the search space with cluster sizes of 10, 20, and 40 as visualized in Figure 3(a) and as described in Section 2.2. For ImageNet-LT and iNaturalist, following previous work [53], we use ResNet-50 and SGD for the lower and upper optimizations. For the learning rate scheduling, we use cosine scheduling starting with learning rate 0.05, and batch size 128. In searching phase, we conduct 150 epoch training with 40 epoch warm-up before the loss function design starts. For the retraining phase, we train for 90 epochs, which is the same as [53] but only due to the lack of training resources available. Due to space limitations, please see supplementary materials for further details.

### Personalized Data Augmentation (PDA)

For PDA, we utilize the AutoAugment [12] policy space for CIFAR datasets, the lower-level optimization trains a ResNet-32 model with standard mini-batch stochastic gradient decent (SGD) using learning rate 0.1, momentum 0.9, and weight decay 1e-4, over 300 epochs. The learning rate decays at epochs 220 and 260 with a factor 0.1. The upper-level hyper-parameter optimization computes the hyper-gradientes via implicit differentiation. Because the hyper-gradient is mostly meaningful when the network achieves near zero loss (Thm 1 of [46]), we start the validation optimization after 120 epochs of the training optimization, using SGD with initial learning rate 0.05, momentum 0.9, and weight decay 1e-4, we follow the same learning rate decay at epoch 220 and 260. For CIFAR10-LT, 20 hyper-parameters are trained, corresponding to \( l_y \) and \( \Delta_y \) of each 10 classes. For CIFAR100-LT, ImageNet-LT, and iNaturalist, we reduce the search space with cluster sizes of 10, 20, and 40 as visualized in Figure 3(a) and as described in Section 2.2. For ImageNet-LT and iNaturalist, following previous work [53], we use ResNet-50 and SGD for the lower and upper optimizations. For the learning rate scheduling, we use cosine scheduling starting with learning rate 0.05, and batch size 128. In searching phase, we conduct 150 epoch training with 40 epoch warm-up before the loss function design starts. For the retraining phase, we train for 90 epochs, which is the same as [53] but only due to the lack of training resources available. Due to space limitations, please see supplementary materials for further details.

### Results and discussion

We compared our methods with the state-of-the-art long-tail learning methods. Table 1 shows the results of our experiments where the design space is parametric CE (2.1). In the first part of the table, we conduct experiments for three baseline methods: normal CE, LDAM [8] and Logit Adjustment loss with temperature parameter \( \tau = 1 \) [53]. The latter choice guarantees Fisher consistency. In the second part of the Table 1, we study Algo. 1 with design spaces \( l, \Delta, \) and \( l&\Delta \). The first version of Algo. 1 in Table 1 tunes the LA loss parameter \( \tau \) where \( l \) is parameterized by a single scalar \( l_y = \tau \log(\pi_y) \). The next three versions of Algo. 1 consider tuning \( l, \Delta, l&\Delta \) respectively (Figure 3b-d shows the evolution of the \( l \) and \( \Delta \) parameters during the optimization). Finally, in last version of Algo. 1, the loss design is initialized with LA loss with \( \tau = 1 \) (rather than balanced CE). The takeaway from these results is that our approach consistently leads to

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR10-LT</th>
<th>CIFAR100-LT</th>
<th>ImageNet-LT</th>
<th>iNaturalist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-Entropy</td>
<td>30.45</td>
<td>62.69</td>
<td>55.47</td>
<td>39.72</td>
</tr>
<tr>
<td>LA loss (( \tau = 1 )) [53]</td>
<td>23.13</td>
<td>58.96</td>
<td>52.46</td>
<td>34.06</td>
</tr>
<tr>
<td>CDT loss [67]</td>
<td>20.73</td>
<td>57.26</td>
<td>53.47</td>
<td>34.46</td>
</tr>
<tr>
<td>AutoBalance: ( \tau ) of LA loss</td>
<td>21.82</td>
<td>58.68</td>
<td>52.39</td>
<td>34.19</td>
</tr>
<tr>
<td>AutoBalance: ( l )</td>
<td>23.02</td>
<td>58.71</td>
<td>52.60</td>
<td>34.35</td>
</tr>
<tr>
<td>AutoBalance: ( \Delta )</td>
<td>22.59</td>
<td>58.40</td>
<td>53.02</td>
<td>34.37</td>
</tr>
<tr>
<td>AutoBalance: ( \Delta &amp; l )</td>
<td>21.39</td>
<td>56.84</td>
<td>51.74</td>
<td>33.41</td>
</tr>
<tr>
<td>AutoBalance: ( \Delta &amp; l, ) LA init</td>
<td>21.15</td>
<td>56.70</td>
<td>50.91</td>
<td>33.25</td>
</tr>
</tbody>
</table>

Table 1: Evaluations of balanced accuracy on long-tailed data. Algo. 1 with \( \Delta \& l \) design space and LA initialization (bottom row) outperforms most of the baselines, across various datasets.

**CEbal.** We consider various designs for \( \ell_{\text{train}} \) such as individually tuning \( w, l, \Delta \) and augmentation. We report the average result of 3 random experiments under this setup.

**Datasets.** We follow previous works [53, 13, 8] to construct long-tailed versions of the datasets. Specifically, for a \( K \)-class dataset, we create a long-tailed dataset by reducing the number of examples per class according to the exponential function \( n_i' = n_i \mu^i \), where \( n_i \) is the original number of examples for class \( i \), \( n_i' \) is the new number of examples per class, and \( \mu < 1 \) is a scaling factor. Then, we define the imbalance factor \( \rho = n_i'/n_K' \), which is the ratio of the number of examples in the largest class \( (n_0') \) to the smallest class \( (n_K') \). For the CIFAR10-LT and CIFAR100-LT dataset, we construct long-tailed versions of the datasets with imbalance factor \( \rho = 100 \). ImageNet-LT contains 115,846 training examples and 1,000 classes, with imbalance factor \( \rho = 256 \). iNaturalist-2018 contains 435,713 images from 8,142 classes, and the imbalance factor is \( \rho = 500 \). These choices follow that of [53]. For all datasets, we split the long-tailed training set into 80% training and 20% validation during the search phase (Figure 1b).

**Implementation.** In both CIFAR datasets, the lower-level optimization trains a ResNet-32 model with standard mini-batch stochastic gradient decent (SGD) using learning rate 0.1, momentum 0.9, and weight decay 1e-4, over 300 epochs. The learning rate decays at epochs 220 and 260 with a factor 0.1. The upper-level hyper-parameter optimization computes the hyper-gradientes via implicit differentiation. Because the hyper-gradient is mostly meaningful when the network achieves near zero loss (Thm 1 of [46]), we start the validation optimization after 120 epochs of the training optimization, using SGD with initial learning rate 0.05, momentum 0.9, and weight decay 1e-4, we follow the same learning rate decay at epoch 220 and 260. For CIFAR10-LT, 20 hyper-parameters are trained, corresponding to \( l_y \) and \( \Delta_y \) of each 10 classes. For CIFAR100-LT, ImageNet-LT, and iNaturalist, we reduce the search space with cluster sizes of 10, 20, and 40 as visualized in Figure 3(a) and as described in Section 2.2. For ImageNet-LT and iNaturalist, following previous work [53], we use ResNet-50 and SGD for the lower and upper optimizations. For the learning rate scheduling, we use cosine scheduling starting with learning rate 0.05, and batch size 128. In searching phase, we conduct 150 epoch training with 40 epoch warm-up before the loss function design starts. For the retraining phase, we train for 90 epochs, which is the same as [53] but only due to the lack of training resources available. Due to space limitations, please see supplementary materials for further details.

**Results and discussion.** We compared our methods with the state-of-the-art long-tail learning methods. Table 1 shows the results of our experiments where the design space is parametric CE (2.1). In the first part of the table, we conduct experiments for three baseline methods: normal CE, LDAM [8] and Logit Adjustment loss with temperature parameter \( \tau = 1 \) [53]. The latter choice guarantees Fisher consistency. In the second part of the Table 1, we study Algo. 1 with design spaces \( l, \Delta, \) and \( l\&\Delta \). The first version of Algo. 1 in Table 1 tunes the LA loss parameter \( \tau \) where \( l \) is parameterized by a single scalar \( l_y = \tau \log(\pi_y) \). The next three versions of Algo. 1 consider tuning \( l, \Delta, l\&\Delta \) respectively (Figure 3b-d shows the evolution of the \( l \) and \( \Delta \) parameters during the optimization). Finally, in last version of Algo. 1, the loss design is initialized with LA loss with \( \tau = 1 \) (rather than balanced CE). The takeaway from these results is that our approach consistently leads to
a superior balanced accuracy objective. That said, tuning the LA loss alone is highly competitive with optimizing $\Delta$ and $l$ alone (in fact, strictly better for CIFAR10-LT, indicating Algo. 1 does not always converge to the optimal design). Importantly, when combining $l$&$\Delta$, our algorithm is able to design a better loss function and outperform all rows across all benchmarks. Finally, when the algorithm further is initialized with LA loss, the performance further improves accuracy, demonstrating that warm-starting with good designs improves performance.

In Table 2, we study the benefits of data augmentation, following our intuitions from Lemma 2. We compare to the differentiable augmentation baseline of MADAO [24] which trains a single policy for the full dataset. PDA is a personalized variation of MADAO and leads to noticeable improvement across all benchmarks (most noticeably in CIFAR10-LT). More importantly, the last two lines of the table demonstrates that PDA can be synergistically combined with the parametric CE (2.1) which leads to further improvements, however, we observe that most of the improvement can be attributed to (2.1).

4 Approaches and Evaluations for Imbalanced Groups

While Section 3 focuses on the fundamental challenge of balanced error minimization, a more ambitious goal is optimizing generic fairness-seeking objectives. In this section, we study accuracy-fairness tradeoffs by examining the group-imbalanced setting.

✓ Setting: Imbalanced groups. For the setting with $G$ groups, dataset is given by $\mathcal{S} = (x_i, y_i, g_i)_{i=1}^n$ where $g_i \in [G]$ is the group-membership. In the fairness literature, groups represent sensitive or protected attributes. For $(x, y, g) \sim \mathcal{D}$, define the group and (class, group) frequencies as

$$
\pi_j = \mathbb{P}_D(g = j), \quad \pi_{k, j} = \mathbb{P}_D(y = k, g = j), \quad \text{for } (k, j) \in [K] \times [G].
$$

The group-imbalance occurs when group or (class, group) frequencies differ, i.e., $\max_j \pi_j \gg \min_j \pi_j$ or $\max_{(k, j)} \pi_{k, j} \gg \min_{(k, j)} \pi_{k, j}$. A typical goal is ensuring that the prediction of the model is independent of these attributes. While many fairness metrics exist, in this work, we focus on the Difference of Equal Opportunity (DEO) [23, 17]. Our evaluations also focus on binary classification (with labels denoted via $\pm$) and two groups ($K = G = 2$). With this setup, the DEO risk is defined as $L_{deo}(f) = |L_{+1}(f) - L_{+2}(f)|$. Here $L_{k,j}(f)$ is the (class, group)-conditional risk evaluated on the conditional distribution of “Class $k$ & Group $j$”. When both classes are equally relevant (rather than $y = +1$ implying a semantically positive outcome), we use the symmetric DEO:

$$
L_{deo}(f) = |L_{+1}(f) - L_{+2}(f)| + |L_{-1}(f) - L_{-2}(f)|. \tag{4.1}
$$

We will study the pareto-frontiers of the DEO (4.1), group-balanced error, and standard error. Here, group-balanced risk is defined as $L_{ba}^{G}(f) = \frac{1}{KG} \sum_{k=1}^{K} \sum_{j=1}^{G} L_{k,j}(f)$. Note that this definition treats each (class, group) pair as its own (sub)group. Throughout, we explicitly set the validation loss to cross-entropy for clarity, thus we use $CE, CE_{ba}^{G}, CE_{deo}$ to refer to $L, L_{ba}^{G}, L_{deo}$.

Validation (upper-level) loss function. In Algo. 1, we set $L_{fair} = (1 - \lambda_{val}) \cdot CE + \lambda_{val} \cdot CE_{deo}$ for varying $0 \leq \lambda_{val} \leq 1$. The parameter $\lambda_{val}$ enables a trade-off between accuracy and fairness. Within $\lambda_{val}$, we use the subscript “val” to highlight the fact that we regularize the validation objective rather than the training objective.

Group-sensitive training loss design. As first proposed in [38], the parametric cross-entropy (CE) can be extended to (class, group) imbalance by extending hyper-parameter $\alpha$ to $[K] \times [G]$ variables $w, l, \Delta \in \mathbb{R}^{[K] \times [G]}$ generalizing (2.1), (2.2). This leads us to the following parametric loss function for group-sensitive classification

$$
\ell_{train}(y, g, f(x); \alpha) = -w_{yg} \log \left( \frac{e^{\alpha(\Delta_{yg})f_{y}(x)+l_{yg}}}{\sum_{k \in [K]} e^{\alpha(\Delta_{kg})f_{k}(x)+l_{kg}}} \right). \tag{4.2}
$$

Here, $w_{yg}$ applies weighted CE, while $\Delta_{yg}$ and $l_{yg}$ are logit adjustments for different (class, groups). This loss function is used throughout the imbalanced groups experiments.

---

Table 2: The evaluations on personalized data optimization.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR10-LT</th>
<th>CIFAR100-LT</th>
<th>ImageNet-LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MADAO [24]</td>
<td>24.39</td>
<td>59.10</td>
<td>55.31</td>
</tr>
<tr>
<td>AutoBalance: PDA</td>
<td>22.53</td>
<td>58.55</td>
<td>54.47</td>
</tr>
<tr>
<td>AutoBalance: $\Delta &amp; l$</td>
<td>21.39</td>
<td>56.84</td>
<td>51.74</td>
</tr>
<tr>
<td>AutoBalance: PDA, $\Delta &amp; l$</td>
<td>20.76</td>
<td>56.49</td>
<td>51.50</td>
</tr>
</tbody>
</table>

---

1This is a generalization to multiple classes of the proposal in [38] for binary group-sensitive classification.
We observe that Algo. 1 significantly Pareto-dominated alternative approaches, for example achieving lower DEO or balanced error for the same standard error. This demonstrates the value of automatic loss function design for a rich class of fairness-seeking objectives.

Next, in Table 3, we solely focus on optimizing the fairness objectives (rather than standard error). Thus, we compare against CE, Group-LA as the baseline approaches as well as blending CE with \( \lambda = 0.1 \). We also compare to the DRO approach of [58]. Finally, we display the outcome of Algo. 1 with \( \lambda_{val} = 0.1 \). While DRO is competitive, similar to Figure 5, our approach outperforms all baselines for all metrics. The improvement is particularly significant when it comes to DEO. Finally, we remark that [38] further proposed combining DRO with the group-adjusted VS-loss for improved performance. We leave the evaluation of AutoBalance for such combinations to future.

**Table 3:** Comparison of fairness metrics for group-imbalanced experiments. The first six rows are baseline training loss functions of the form \( \min \frac{1}{|G|} \sum_{g \in G} \mathbb{E}_{x \sim G} \mathcal{L}(f(x), y) \). Next, in Table 3, we solely focus on optimizing the fairness objectives (rather than standard error). Finally, we display the outcome of Algo. 1 with \( \lambda_{val} = 0.1 \) in Algo. 1.

<table>
<thead>
<tr>
<th>Loss function</th>
<th>Balanced Error</th>
<th>Worst (class, group) error</th>
<th>DEO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross entropy (CE)</td>
<td>23.38</td>
<td>43.25</td>
<td>33.75</td>
</tr>
<tr>
<td>CE(_{bal}^g)</td>
<td>20.83</td>
<td>36.67</td>
<td>20.25</td>
</tr>
<tr>
<td>Group-LA loss</td>
<td>22.83</td>
<td>40.50</td>
<td>29.33</td>
</tr>
<tr>
<td>CE(_{deo})</td>
<td>19.29</td>
<td>35.17</td>
<td>25.25</td>
</tr>
<tr>
<td>0.1 \cdot CE + 0.9 \cdot CE(_{deo}) (( \lambda = 0.1 ))</td>
<td>20.06</td>
<td>31.67</td>
<td>26.25</td>
</tr>
<tr>
<td>DRO [58]</td>
<td>16.47</td>
<td>32.67</td>
<td>6.91</td>
</tr>
<tr>
<td>AutoBalance: ( \mathcal{L}<em>{\text{fair}} ) with ( \lambda</em>{val} = 0.1 )</td>
<td>15.13</td>
<td>30.33</td>
<td>4.25</td>
</tr>
</tbody>
</table>

**Baselines.** We will compare Algo. 1 with training loss functions parameterized via \((1 - \lambda) \cdot CE + \lambda \cdot \mathcal{L}_{\text{reg}}\). Here \( \mathcal{L}_{\text{reg}} \) is a fairness-promoting regularization. Specifically, as displayed in Table 3 and Figure 5, we will set \( \mathcal{L}_{\text{reg}} \) to be \( CE_{bal}^g\), CE\(_{deo}\), and Group LA. “Group LA” is a natural generalization of the LA loss to group-sensitive setting: it chooses weights \( w_g = 1/\bar{\pi}_g \) to balance group frequencies and then applies logit-adjustment with \( \tau = 1 \) over the classes conditioned on the group-membership.

| Datasets.** We experiment with the modified Waterbird dataset [58]. The goal is to correctly classify the bird type despite the spurious correlations due to the image background. The distribution of the original data is as follows. The binary classes \( k \in \{-, +\} \) correspond to \{waterbird, landbird\}, and the groups \( |G| = \{1, 2\} \) correspond to \{land background, water background\}. The fraction of data in each (class, group) pair is \( \pi_{-2} = 0.22, \pi_{-1} = 0.012, \pi_{+,2} = 0.038, \) and \( \pi_{+,1} = 0.75 \). The landbird on the water background \( \{(+,2)\} \) and the waterbird on the land background \( \{(-,1)\} \) are minority subgroups within their respective classes. The test set, following [58], has equally allocated bird types on different backgrounds, i.e., \( \pi_{+,j} = 0.25 \). As the test dataset is balanced, the standard classification error \( \mathcal{E}(f) \) is defined to be the weighted error \( \mathcal{E}(f) = \pi_{y,g} \mathcal{E}_{yg}(f) \).

**Implementation.** We follow the feature extraction method from [58], where \( x \) are 512-dimensional ResNet18 features. When using Algo. 1, we split the original training data into 50% training and 50% validation. The search phase uses 150 epochs of warm up followed by 350 epochs of bilevel optimization. The remaining implementation details are similar to Section 3.

**Results and Discussion.** We consider various fairness-related metrics, including the worst (class, group) error, DEO \( \mathcal{E}_{deo}(f) \) and the balanced error \( \mathcal{E}_{bal}^g(f) \). We seek to understand whether AutoBalance algorithm can improve performance on the test set compared to the baseline training loss functions of the form \((1 - \lambda) \cdot CE + \lambda \cdot \mathcal{L}_{\text{reg}}\). In Figure 5, we show the influence of the parameter \( \lambda \) where \( \mathcal{L}_{\text{reg}} \) is chosen to be CE\(_{deo}\), CE\(_{bal}^g\), or Group-LA (each point on the plot represents a different \( \lambda \) value). As we sweep across values of \( \lambda \), there arises a tradeoff between standard classification error \( \mathcal{E}(f) \) and the fairness metrics. We observe that Algo. 1 significantly Pareto-dominates alternative approaches, for example achieving lower DEO or balanced error for the same standard error. This demonstrates the value of automatic loss function design for a rich class of fairness-seeking objectives.
5 Related Work

Our work relates to imbalanced classification, fairness, bilevel optimization, and data augmentation. Below we focus on the former three and defer the extended discussion to the supplementary.

Long-tailed learning. Learning with long-tailed data has received substantial interest historically, with classical methods focusing on designing sampling strategies, such as over- or under-sampling [40, 60, 9, 41, 1, 71, 57, 68, 5, 50]. Several re-weighting schemes [54, 52, 25, 13, 6, 34] have been proposed to adjust weights of different classes or samples during training. Another line of work [70, 37, 53] focuses on post-hoc correction. More recently, several works [44, 35, 16, 34, 8, 13, 53, 67] develop more refined class-balanced loss functions (e.g. (2.1)) that better adapt to the training data. In addition, several works [30, 74] point out that separating the representation learning and class balancing can lead to improvements. In this work, our approach is in the vein of class-balanced losses; however, rather than fixing a balanced loss function (e.g. based on the class probabilities in the training dataset), we employ our Algorithm 1 to automatically guide the loss design.

Group-sensitive and Fair Learning. Group-sensitive learning aims to ensure fairness in the presence of under-represented groups (e.g., gender, race). [7, 23, 69, 64] propose several fairness metrics as well as insightful methodologies. A line of research [4, 19] optimize the worst-case loss over the test distribution and further applications motivate (label, group) metrics such as equality of opportunity [23, 17] (also recall DEO (4.1)). [58] discusses group-sensitive learning in an over-parameterized regime and proposes that strong regularization ensures fairness. Closer to our work, [58, 38] also study Waterbirds dataset. Compared to the regularization-based approach of [58], we explore a parametric loss design (inspired by [38]) to optimize fairness-risk over validation. [17] proposes methods and statistical guarantees for fair empirical risk minimization. A key observation of our work is that, such guarantees based on training-only optimization can be vacuous in the overparameterized regime. Thus, using train-validation split (e.g. our Algo 1) is critical for optimizing fairness metrics more reliably. This is verified by the effectiveness of our approach in the evaluations of Section 4.

Bilevel Optimization. Classical approaches [61] for hyper-parameter optimization are typically based on derivative-free schemes, including random search [65] and reinforcement learning [76, 3, 66, 72]. Recently, a growing line of works focus on differentiable algorithms that are often faster and can scale up to millions of parameters [46, 49, 59, 28, 48]. These techniques [45, 36, 75, 47] have shown significant success in neural architecture search, learning rate scheduling, regularization, etc. They are typically formulated as a bilevel optimization problem: the upper and lower optimizations minimize the validation and training losses, respectively. Some theoretical guarantees (albeit restrictive) are also available [11, 21, 2, 55], Different from these, our work focuses on principled design of training loss function to optimize fairness-seeking objectives for imbalanced data. Here, a key algorithmic distinction (e.g. compared to architecture search) is that, our loss function design is only used during optimization and not during inference. This leads to a more sophisticated hyper-gradient and necessitates additional measures to ensure stability of our approach (see Algo 1).

6 Conclusions and Future Directions

This work provides an optimization-based approach to automatically design loss functions to address imbalanced learning problems. Our algorithm consistently outperforms, or is at least competitive with, the state-of-the-art approaches for optimizing balanced accuracy. Importantly, our approach is not restricted to imbalanced classes or specific objectives, and can achieve good tradeoffs between (accuracy, fairness) on the Pareto frontier. We also provide theoretical insights on certain algorithmic aspects including loss function design, data augmentation, and train-validation split.

Potential Limitations, Negative Societal Impacts, & Precautions: Our algorithmic approach can be considered within the realm of automated machine learning literature (AutoML) [27]. AutoML algorithms often optimize the model performance, thus reducing the need for engineering expertise at the expense of increased computational cost and increased carbon footprint. For instance, our procedure is computationally more intensive compared to the theory-inspired loss function prescriptions of [53, 8]. A related limitation is that Algo. 1 can be brittle in extremely imbalanced scenarios with very few samples per class. We took the several steps to help mitigate such issues: first, our algorithm is initialized with a Bayes consistent loss function to provide a warm-start (such as the proposal of [53]). Second, to improve generalization and avoid overfitting to validation, we reduce the hyperparameter search space by grouping the classes with similar frequencies. Finally, evaluations show that the designed loss functions are interpretable (Fig. 3(b,c,d)) and are consistent with theoretical intuitions.
Acknowledgments
This work is supported in part by the National Science Foundation under grants CCF-2046816, CNS-1932254, CCF-2009030, HDR-1934641, CNS-1942700 and by the Army Research Office under grant W911NF-21-1-0312.

References


