Feature Generation for Long-tail Classification

Rahul Vigneswaran¹, Marc T. Law², Vineeth N. Balasubramanian¹, Makarand Tapaswi³
¹Indian Institute of Technology, Hyderabad ²NVIDIA ³IIIT Hyderabad, India

ABSTRACT
The visual world naturally exhibits an imbalance in the number of object or scene instances resulting in a long-tailed distribution. This imbalance poses significant challenges for classification models based on deep learning. Oversampling instances of the tail classes attempts to solve this imbalance. However, the limited visual diversity results in a network with poor representation ability. A simple counter to this is decoupling the representation and classifier networks and using oversampling only to train the classifier.

In this paper, instead of repeatedly re-sampling the same image (and thereby features), we explore a direction that attempts to generate meaningful features by estimating the tail category’s distribution. Inspired by ideas from recent work on few-shot learning [53], we create calibrated distributions to sample additional features that are subsequently used to train the classifier. Through several experiments on the CIFAR-100-LT (long-tail) dataset with varying imbalance factors and on mini-ImageNet-LT (long-tail), we show the efficacy of our approach and establish a new state-of-the-art. We also present a qualitative analysis of generated features using t-SNE visualizations and analyze the nearest neighbors used to calibrate the tail class distributions. Our code is available at https://github.com/rahulvigneswaran/TailCalibX.

CCS CONCEPTS
• Computing methodologies → Supervised learning by classification.

KEYWORDS
Long-tail classification, Feature generation

ACM Reference Format:
Rahul Vigneswaran, Marc T. Law, Vineeth N. Balasubramanian, Makarand Tapaswi. 2021. Feature Generation for Long-tail Classification. In Proceedings of 12th Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP’21). ACM, New York, NY, USA, Article 43, 9 pages. https://doi.org/10.1145/3490035.3490300

1 INTRODUCTION
Modern machine learning is driven by large scale datasets, employed in both supervised [11, 33, 36] and self-supervised [31, 34, 37, 39] scenarios. However, creation of these labeled datasets is a challenging and costly affair [1] and may also lead to unforeseen biases [41]. Often, these datasets are created by querying images through a search engine [33, 36], followed by post-processing and “cleaning” to ensure a balanced distribution that has an (approximately) equal number of instances for each category. However, the world we live in is naturally long-tail, and like the language modality, even visual data follows the Zipf’s law [44, 57]. This is easily illustrated through examples we encounter in our daily lives: people living in a city are more likely to see multiple instances and a large diversity of cars than elephants for transportation, and tables and chairs than tree stumps as furniture. This natural distribution of categories is reflected in the datasets that are collected through community efforts, e.g., iNaturalist [44] that features a large image collection of biodiversity; or annotations for a collection of randomly sampled raw data – action labels from the Atomic Visual Actions dataset [13]; or interaction and relationship labels from movie datasets [46].

Modern deep learning methods perform well on balanced distributions and have even shown super-human performance for some datasets and tasks in diverse domains including image classification [14] and language understanding [48]. This has led to a steadily growing interest in adapting methods to work well with few training samples – few-shot learning [4, 10, 29, 49, 51] or naturally occurring long-tail distributions [17, 20, 24, 40, 56]. Interestingly, it is observed that popular techniques used in shallow learning (e.g., SVMs) such as loss re-weighting or balanced sampling [19] do not perform well when they are applied with deep neural networks,
as they may hurt the representation learning [20, 56]. Instead, de-
coupling the feature learning (i.e. CNN backbone) from the classifier
(i.e. final linear layers) is found to be necessary and useful [20].

In this work, we are interested in answering the following ques-
tion. Can we learn to generate feature representations for the im-
poverished tail classes, instead of requiring re-weighting (e.g. [56])
or multi-network distillation (e.g. [17]) strategies? Inspired by the
recent success of distribution calibration on few-shot learning [53],
we estimate and calibrate the feature distribution of tail classes to
sample additional features that are in turn used to train a classifier
- we term this as TailCalibration (see Fig. 1). Note that our work
is different from [5, 9, 21] that attempt to reconstruct or generate
images for the tail class; or from [45, 54] that use convex combina-
tions of input samples or features and their labels. TailCalibration
is agnostic to the training approach of the backbone, and therefore,
adopting better classification setups (e.g. CosineCE [28]) or using
distillation (e.g. CBD) can further improve overall performance.

Our contribution can be summarized as follows: We explore
feature generation as a means to address the challenges of long-tail
classification, and show that TailCalibration, an adapted version of
[53], is an effective strategy to generate meaningful additional
features (Sec. 3). We empirically validate our approach through
multiple ablation studies on the CIFAR-100-LT achieving the state-
of-the-art performance (Sec. 4). Our analysis also holds for a long-
tailed version of a similarly sized dataset called mini-ImageNet-
LT [47] that had been introduced for few-shot learning. We also
present a qualitative analysis of generated features using t-SNE
visualizations and analyze the category-level nearest neighbors
used to calibrate the tail class distributions.

2 RELATED WORK

We discuss and contrast related work on long-tail classification
with our proposed approach. While imbalanced learning has had a
long history, we focus primarily on modern techniques applied to
deep learning. In general, we can group related works in 4 broad
categories: (i) modifications to the loss function or re-weighting
of samples; (ii) decoupling the learning process of the representation
and classifier; (iii) using distillation or multiple experts; and (iv)
perhaps closest to our work, ideas related to sample or feature
generation.

2.1 Re-weighting or Adapting the Loss Function

Class re-weighting or balanced sampling are popular approaches to
deal with long-tail datasets especially in shallow architectures [19].
However, with deep learning, as the same loss function trains both
the representation network and the classifier, the above ideas are
not directly applicable.

The cross-entropy (CE) loss is modified to down-weight easier
examples while focusing on harder ones [24], or to include the
effective number of training samples via a re-weighting term [8].
As an alternative to CE, there has also been work on deriving a
class-specific margin parameter to be inversely proportional to
the fourth root of number of samples for a margin-based loss [3].
Long-tail classification and label corruption are also attempted
simultaneously through meta-learning networks for re-weighting
instances [35, 38]. Recently, a simple logit adjustment that can be
applied post-hoc to trained models [30] is proposed as a means to
unify many of the above approaches.

A few different perspectives applied to long-tail classification
include viewing it as a domain adaptation task (target shift) [16,
18], or as a causal framework where stochastic gradient descent’s
momentum term is treated as a confounder [40].

Our work does not belong to this category and can be thought of
as orthogonal to any of the techniques above.

2.2 Decoupling Representation and Classifier

One of the key challenges of deep learning on long-tail datasets
is exposure by [20, 56]. Both works demonstrate that (i) represen-
tation learning suffers when applying balancing or re-weighting
techniques on end-to-end training paradigms, and (ii) classifier per-
f ormance of tail classes is adversely impacted when using standard
instance sampling. As remedies, a two stage approach is suggested.
In the first stage, usual end-to-end training is performed using in-
stance sampling. In the second stage, the classifier is decoupled from
the network and is trained with balanced sampling [20]. While this
two-stage strategy improves the performance, [55] discovers that
this may lead to domain shift between the representation and the
classifier parts when seen from the perspective of transfer learning
and proposes MiSLAS - a batch normalization based trick to resolve
this. On similar lines, a dual network with shared backbone is pro-
based on [56] with an adaptive trade-off parameter that balances
the two objectives.

Our work is related to the above - we also decouple the training
of the backbone from the classifier. The key difference is that instead
of balanced sampling, we attempt to generate meaningful features
for the tail classes that are subsequently used to train the classifier.

2.3 Distilling Information across Networks

An alternative approach to long-tail classification is using (multiple)
teachers to distill information into a student network. One idea
here involves learning separate classifiers for sets of categories such
that the segregated imbalance factor is reduced before combining
them [52]. Along those lines, RIDE [50] trains multiple experts with
shared earlier layers and routes the inference through a subset of
trained experts to improve tail class performance. These subset of
experts can be self-distilled from the a larger subset of experts for
further improvement. An alternative idea considers an ensemble of
teachers trained on instance sampling that are used to uphold the
quality of the representation network while training the student
on a balanced sampling schedule [17].

Our strategy is orthogonal to this approach and we show that
applying TailCalibration on the distilled model from [17] further
improves performance.

2.4 Instance or Feature Generation

The final set of approaches address the challenges of long-tail class-
ification through generation or augmentation of samples or features.
Among sample generation, SMOTE [5] is an old but popular choice
that generates new instances through interpolation between sam-
ples of the same (often, the tail) class. Parallel to our work, SMOTE
has been extended to deep learning where an encoder-decoder
We present details of our proposed method, TailCalibration, that allows models to learn from convex combinations of not only input samples, but also their labels [54]. Specifically designed for visual long-tail classification, M2m translation transfers the diversity of samples from the head class to the tail through gradient updates, resulting in minor visual transformations that lead to significant impact on classification scores [21]. Similar to M2m, [7] explicitly models class-specific and class-generic features which are then mixed together to transfer the sample diversity from the head class to the tail.

While mixup is applied on the input space, manifold mixup is applied on the semantic (feature) space [45] and is effective at generating convex feature combinations. Specific data augmentation strategies relevant for long-tail classification transfer the implicit knowledge of features from the head classes that exhibit a high variance, to the tail classes that lack intra-class diversity [25]. This is done by constructing a feature cloud around the existing tail data points to match the head class variance. A recent work MODALS [6] suggests multiple modality agnostic feature augmentation strategies based on hard example interpolation, extrapolation, and Gaussian noise around the samples.

Our work is related to these ideas, but differs in the actual method for feature generation. By calibrating distributions we expect to learn from the diversity of head classes and generate features for tail classes to balance the inputs seen by the classifier.

3 METHOD
We present details of our proposed method, TailCalibration, that generates additional features for the tail classes. We start by formulating the long-tail classification task through some notation and a discussion of multiple strategies to train the backbone network.

3.1 Problem Definition
We address $K$ class supervised classification on a training dataset $D = \{(x_i, y_i)\}_{i=1}^N$ with $N$ samples. $x_i$ represents the $i$th image and $y_i$ is the one-hot encoding of the category label $c_i \in \{1, \ldots, K\}$. For simplicity, we denote the set of samples belonging to category $k$ by $D_k$, and denote its cardinality by $N_k = |D_k|$. A long-tail setup can be defined by ordering the number of samples per category, i.e., $N_1 \geq N_2 \geq \ldots \geq N_K$ (without loss of generality), such that $\sum_k N_k = N$.

The imbalance factor of the dataset is indicated as the ratio of samples in the head to tail class, $N_1/N_K$, where a higher imbalance often translates to worse performance on the tail categories.

We train a network $\Phi$ consisting of two components: (i) a backbone or representation network (CNN for images) that translates an image to a feature representation $f(x_i) = z_i$ where $z_i \in \mathbb{R}^D$, and (ii) a classifier $W \in \mathbb{R}^{K \times D}$ that predicts the category specific scores (logits) $g(z_i) = s_i$. When not specified otherwise, we assume $s_i = g(z_i) = Wz_i$. We ignore writing the bias term of the linear layer for brevity.

3.2 Training the Backbone
Our approach on generating features is independent from the manner in which the representation network is trained. Below, we present three ways of obtaining a trained backbone network $f$.

We consider a batch-wise training approach with instance sampling, where each sample $x_i$ has equal probability of being considered as part of a mini-batch $B$. Specifically, the parameters of the backbone $f$ and the classifier $g$ are trained end-to-end using the cross-entropy (CE) loss:

$$L_{CE} = -\sum_i y_i \log(p_i),$$

where $p_i$ is a vector of probabilities obtained by transforming the logits $s_i$ through the softmax operation.

CosineCE. As an alternative to the standard dot-product classifier, a cosine normalization based classifier is proposed to reduce the variance of the output neuron [28]. The key difference lies in the computation of the dot product - the feature vector $x_i$ and each row (category $k$) of the classifier, $w_k$, are $\ell_2$-normalized prior to computing the dot product. The logit score for sample $x_i$ and category $k$ is computed as follows:

$$s^\text{CosCE}_{ik} = \gamma \frac{z_i^\top w_k}{\|z_i\|_2 \|w_k\|_2}$$

where $\gamma \in \mathbb{R}^+$ is a positive normalization constant that ensures that $s^\text{CosCE}_{ik} \in [-\gamma, \gamma]$. We treat $\gamma$ as a learnable parameter and use the softplus operator to ensure that it remains positive. We recall that the operator is defined as $\forall x \in \mathbb{R}$, $\text{softplus}(x) = \log(1 + e^x)$. The loss function used to train the model remains unchanged from the above Eq. (1).

This formulation has also been adopted by recent works on few-shot learning [12] and long-tail classification [17]. In particular, the cosine similarity based score computation embeds all samples close to a single point on the hypersphere (i.e. a $(D-1)$-sphere embedded in $\mathbb{R}^D$) indicated by the classifier vector. We believe that this makes the distribution of features on the hypersphere closely resemble a Gaussian distribution or a von Mises-Fisher distribution [2]. We will show the benefits of using CosineCE through our ablation studies.

Class-balanced Distillation (CBD). Parallel to our work, Iscen et al. [17] show the use of teacher-student distillation with the goal of improving both the backbone representation and the classifier. We briefly summarize their approach. First, a teacher network $\Phi_T : (f_T, g_T)$ is trained using standard instance sampling and the CosineCE classifier described above. Then, a student network $\Phi_S : (f_S, g_S)$ is trained with balanced sampling along with distillation, i.e. the intermediate feature representation $f_S(x_i)$ is constrained to be similar to $f_T(x_i)$, while harnessing the benefits of balanced sampling for the classifier $g_S$. We decouple the backbone network trained through a combination of the classification and distillation losses, and show that TailCalibration can use these improvements to obtain additional performance gains.

3.3 TailCalibration
The primary objective of our work is to generate additional features $x^*$ such that they can be used to balance the data that is used to train the classifier. Distribution calibration (DC) was proposed recently in the context of few-shot learning: samples are generated for the few-shot classes by relying on statistics of the base classes [53]. We investigate the applicability of DC in the context of long-tail classification, and present the generation process below.
Figure 2: We visualize the process of feature generation. We first estimate the class statistics, and apply the Tukey transformation to convert the data into a distribution closer to a Gaussian. We find the nearest neighbor categories for each instance, calibrate the tail class distribution, and sample from this to generate additional features. The process is repeated until all categories have balanced number of features, which are used as training data for the classifier.

Our approach can be simplified into three steps, also illustrated in Fig. 2. (i) We first estimate the distribution of each category based on a Gaussian assumption; (ii) Categorical neighbors of each instance in the tail classes are used to create a calibrated distribution; and (iii) We sample several new features from these distributions to balance the training data seen by a classifier.

All operations below are in the feature space. Given a trained backbone (discussed in Sec. 3.2), we first precompute feature representations for the entire dataset. These features of true samples are denoted by \( \mathcal{F} = \{ z_i \}_{i=1}^{N} \), \( \mathcal{F}_k \) denotes features of images corresponding to the subset \( D_k \), or belonging to the category \( k \).

Due to the inherent randomness in sampling from a distribution, we can generate features multiple times and use them as a fresh dataset from which the classifier can learn something new in each epoch.

**Class statistics.** We start by computing the statistics for each category, assuming that the feature distribution is Gaussian. Note that this assumption may be particularly reasonable when using a backbone trained with the CosineCE loss function.

\[
\mu_k = \frac{1}{N_k} \sum_{i \in F_k} z_i \quad \text{and} \quad \Sigma_k = \frac{1}{N_k - 1} \sum_{i \in \mathcal{F}_k} (z_i - \mu_k)(z_i - \mu_k)^T .
\]

where \( \mu_k \in \mathbb{R}^D \) and \( \Sigma_k \in \mathbb{R}^{D \times D} \) denote the mean and full covariance of the Gaussian distribution for category \( k \).

**Tukey’s Ladder of Powers transformation.** Sometimes referred to as the Bulging rule, this transformation helps change the shape of a skewed distribution so that it becomes closer to a Normal distribution [42]. In particular, it is applied to each dimension of the feature as follows:

\[
\tilde{z}_i = \begin{cases} 
\frac{z_i^\lambda}{\log(z_i)} & \text{if } \lambda \neq 0 \\
\log(z_i) & \text{otherwise}.
\end{cases}
\]

where \( \lambda > 0 \) is a hyperparameter of our model and \( z_i^\lambda \) raises each element of \( z_i \) to the power \( \lambda \). In fact, \( \lambda = 1 \) is found to perform well for most of our experiments.

**Calibration and generation.** For each class, we sample \( N_1 - N_k \) additional features, such that the resulting feature dataset is completely balanced and all classes have \( N_1 \) instances. Sampling is performed based on an instance specific calibrated distribution. Specifically, each \( z_{ik} \) \((i^{th} \text{ feature from category } k)\) is responsible for generating \( \lfloor N_1/N_k - 1 \rfloor \) features where \( \lfloor x \rfloor \) rounds \( x \) to the nearest integer greater than or equal to \( x \). As \( N_1/N_k \) may be a fraction, we randomly choose an appropriate number of \( z_{ik} \) so as to obtain a total of \( N_1 \) samples.

Next, we compute the distances between class means and the selected feature. For each feature \( z_i \), we first compute the distance between the instance and category means as

\[
\forall k, d_{ik} = ||z_i - \mu_k||_2.
\]

We identify the set of \( M \) category indices that are neighbors \( N_i \) with smallest distance \( d_{ik} \). Note that the distance computation uses the Tukey transformed feature \( \tilde{z}_i \).

The calibrated distribution is obtained as

\[
\tilde{\mu}_i = \frac{1}{M+1} \left( \sum_{k \in N_i} \mu_k + \tilde{z}_i \right) \quad \text{and} \quad \tilde{\Sigma}_i = \frac{1}{M} \left( \sum_{k \in N_i} \Sigma_k \right) + \alpha ,
\]

Plugging these values into the above formulas gives the transformed feature as needed for the classifier.
where $\alpha$ is an optional constant hyper-parameter to increase the spread of the calibrated distribution. We found that $\alpha = 0$ works reasonably well for multiple experiments.

**Sampling.** We initialize a multi-variate Gaussian distribution\(^1\) for each instance $z_i$, and generate new features with the same associated class label as $c_i$. We denote the generated features for category $k$ as $F_k$, and together, $|F_k| = N_k$. This combined set of features is generated for all categories and used to train the classifier $g$.

$\ell_2$-normalized calibration. For the CosineCE backbone, we $\ell_2$-normalize the features $z_i$ before feeding them to the TailCalibration pipeline, both for gathering statistics and Tukey transformation.

4 EXPERIMENTS

In this section, we start by presenting a brief overview of the dataset and implementation details for our methods. We present some ablation studies comparing different aspects of the model, and finally perform a thorough comparison of our work against state-of-the-art approaches. We also include some analysis to obtain a better understanding of the feature generation process.

4.1 Datasets and Experimental Setup

We benchmark our proposed method across two datasets, CIFAR-100-LT and mini-ImageNet-LT.

**CIFAR-100-LT.** CIFAR-100 is a balanced dataset containing 60K images from 100 categories, with 50K in train and 10K in validation. We use the synthetically created long-tail variants that have also been used by previous works [3, 56]. There are three versions of the CIFAR-100-LT dataset, with the imbalance factor $(N_i/N_K)$ of $\{10, 50, 100\}$. An exponential decay is used for determining $N_k$ in between. Note that a higher imbalance factor mimics a stronger long-tail problem and is typically more challenging - we will see this through the performance of baseline models.

As the synthetically created variants involve randomly selecting images, all experiments are replicated with 3 seeds where different sets of images are selected. The average performance over all seeds is reported when not mentioned otherwise. As we will see in the ablation studies, this random sub-sampling often leads to a high variance, but we observe consistent improvements by using our approach. For evaluation, we use the entire balanced validation set of 10K images, 100 samples for each of the 100 categories.

**mini-ImageNet-LT.** mini-ImageNet was proposed by [47] for few-shot learning evaluation, in an attempt to have a dataset like ImageNet while requiring fewer resources. Similar to the statistics for CIFAR-100-LT with an imbalance factor of 100, we construct a long-tailed variant of mini-ImageNet that features all the 100 classes and an imbalanced training set with $N_1 = 500$ and $N_K = 5$ images. For evaluation, both the validation and test sets are balanced and contain 10K images, 100 samples for each of the 100 categories.

We report performance as accuracy. Note that average per-sample accuracy is equivalent to average per-class accuracy due to the balanced evaluation sets in both datasets.

\(^1\)using PyTorch’s MultivariateNormal distribution.

### Table 1: Hyperparameters used in TailCalibration feature generation across various imbalance ratios and datasets.

| Hyperparameters | CIFAR-100-LT Imbalance Ratios | mini-ImageNet-LT Imbalance Ratios |
|-----------------|-------------------------------|-----------------------------------|
| $\lambda$ (Tukey value) | 1.0 | 0.9 | 0.9 | 1.0 |
| $M$ (Number of categories) | 3 | 2 | 2 | 2 |
| $\alpha$ (Spread of generated features) | 0.0 | 0.2 | 0.0 | 0.1 |

### Table 2: Ablation study comparing the impact of $\ell_2$-normalization of features prior to TailCalibration. Backbone: CosineCE, Feature generation: TailCalib.

| Method | Imbalance Ratios |
|--------|------------------|
| without $\ell_2$-normalization | 42.62 | 49.17 | 58.12 |
| with $\ell_2$-normalization | 43.03 | 49.18 | 58.60 |

4.2 Implementation details

**Implementation details for CIFAR100-LT.** We follow a similar setup to [56]: the backbone is a ResNet-32 [15] trained by minibatch stochastic gradient descent (SGD) with a momentum of 0.9 and a batch size of 128. For a fair comparison, we tune the other parameters to match the baseline accuracy of the rest of the reported works. The learning rate is decayed by a cosine scheduler [27] from 0.2 to 0.0 in 150 epochs, and a weight decay of $5 \times 10^{-5}$ is used. We train our models on a single NVIDIA 1080Ti GPU.

**Implementation details for mini-ImageNet-LT.** We follow a similar setup to [20, 40]: the backbone is a ResNeXt-50 trained by minibatch stochastic gradient descent (SGD) with a momentum of 0.9 for 111 epochs on a batch size of 512. Through hyperparameter tuning, we found that weight decay of $5 \times 10^{-4}$ with a constant learning rate of 0.01 provides a better performing baseline. We train all the models on a NVIDIA Tesla P100 GPU.

For the TailCalibration feature generation, we use the hyperparameters as stated in Table 1. Subsequently, the classifier is re-trained with a constant learning rate of 0.001 for CIFAR-100-LT and 0.01 for mini-ImageNet-LT. While the weight decay for retraining the classifier remains the same as the backbone for CIFAR-100-LT, we found that decreasing the weight decay from $5 \times 10^{-4}$ to $5 \times 10^{-5}$ improves performance for mini-ImageNet-LT.

4.3 Ablation studies on CIFAR-100-LT

We present a few ablation studies to compare various aspects of our proposed method. Note that we do all the necessary ablations only with CIFAR-100-LT and use the findings to guide our decisions on mini-ImageNet-LT.

$\ell_2$-normalization. As indicated in the last paragraph of Sec. 3.3, when using CosineCE, we observed that it is better to $\ell_2$-normalize the feature vectors $z_i$ before and after the generation. We show the impact of this normalization in Table 2. Imbalance factors 100
Table 3: Ablation study comparing the impact of backbones: CE vs. CosineCE vs. CBD. We also see that generating features multiple times can be beneficial for training: TailCalib vs. TailCalibX.

| Backbone | Feature Generation | Imbalance Ratios |
|----------|--------------------|------------------|
|          |                    | 100   | 50   | 10   |
| CE       | TailCalib          | 42.28 | 46.37| 56.81|
|          | -                  | 39.89 | 44.88| 57.00|
| CosineCE | TailCalib          | 41.44 | 45.72| 58.58|
|          | TailCalibX         | 44.44 | 49.80| 59.73|
| CBD [17] | TailCalib          | 46.22 | 50.87| 60.78|
|          | TailCalibX         | 46.59 | 50.90| 61.13|

and 10 see a small but consistent improvement across all three seeds, pointing to the relevance of this modification: +0.21%, +0.64%, and +0.38% for imbalance 100 and +0.77%, +0.4%, and +0.29% for imbalance 10. For imbalance 50, we see a small improvement for 2 of 3 seeds: +0.12%, +0.09%, and -0.19%, resulting in an overall negligible performance difference.

Backbones. We presented three main strategies to train our backbone in Sec. 3.2. Table 3 reports performance when applying TailCalibration (abbreviated as TailCalib) for features precomputed from various backbones (averaged across 3 seeds). Firstly, note that CosineCE outperforms CE, while CBD with teacher-student distillation outperforms both. TailCalibration shows a consistent improvement over backbones, especially for the higher imbalance factors of 50 and 100.

Training with multiple rounds of generation. As the feature generation process is quite fast, we can afford to generate samples on-the-fly for each epoch during the training of the classifier. In particular, we propose TailCalibX, denoting that TailCalib is employed multiple times. Specifically, we generate a set of features once every epoch and use them to train the classifier. We show the results for this experiment in Table 3. TailCalibX provides incremental performance boosts over TailCalib.

Trends across different seeds. The variation across seeds is typically high as changing the seed not only changes the random initialization of the base network (this has a small impact), but also leads to the selection of a different subset of CIFAR-100 images for creating the synthetically imbalanced training data (this has a large impact). This is especially true for imbalance factor 50 and 10; we suspect that as there are very few training samples in the tail categories for imbalance factor 100, samples are selected does not matter much. For example, the accuracy of the CE baseline across 3 seeds is (43.8%, 44.8%, 46.1%) for imbalance factor 50.

Fig. 3 shows the absolute percentage points performance improvement over the specific backbone for each seed. We observe consistent performance improvement over progressively harder backbones by applying TailCalib or TailCalibX across various seeds.

Table 4: Comparison of our approach (TailCalib, TailCalibX) against previous works on CIFAR-100-LT. Methods with a * indicate results produced by our re-implementations, please see the text for details about the implementation. Best results are highlighted in bold, next best and notable results with underline.

| Type          | Method               | Imbalance Ratios |
|---------------|----------------------|------------------|
|               |                      | 100   | 50   | 10   |
| Baseline      | *CE                  | 39.89 | 44.88| 57.00|
|               | *CosineCE            | 41.44 | 45.72| 58.58|
|               | Focal Loss [24]      | 38.41 | 44.32| 55.78|
|               | Class-Balanced Focal [8] | 39.60 | 45.32| 57.99|
|               | L2RW [35]            | 38.90 | 46.83| 52.12|
| Loss or Re-weighting | Meta-Weight Net [38] | 41.61 | 45.66| 58.91|
|               | Domain Adaptation [18] | 39.31 | 48.53| 59.58|
|               | LDAM-DRW [3]         | 42.04 | -    | 58.71|
|               | Logit adjustment [30] | 43.89 | -    | -    |
|               | GBM [40]             | 44.1  | 50.3 | 59.6 |
| Decouple      | *cRT [20]            | 42.63 | 47.26| 57.61|
|               | BBN [56]             | 42.56 | 47.02| 59.12|
| Distillation  | LFAE [52]            | 42.3  | -    | -    |
|               | *CBD [17]            | 44.83 | 49.19| 60.85|
| Generation    | M2M [21]             | 42.9  | -    | 58.2 |
|               | mixup [54]           | 39.54 | 44.99| 58.02|
|               | Manifold mixup [45]  | 38.25 | 43.09| 56.55|
|               | *MODALS [6]          | 42.813| 47.57| 58.53|
|               | *CosineCE + TailCalib| 43.03 | 49.18| 58.60|
|               | *CosineCE + TailCalibX| 44.44 | 49.80| 59.73|
|               | CBD + TailCalibX     | 46.59 | 50.90| 61.13|

We can also see that while TailCalib may fail to provide performance improvements for low imbalance factors (CosineCE and CBD, imbalance 10), TailCalibX always provides a small boost. This indicates that the feature generation process may not be very reliable for datasets with a small long-tail effect, but the classifier can still extract meaningful information from multiple rounds of generation.

4.4 Comparison to state-of-the-art
We compare our proposed approach against several previous works grouping them in a manner similar to our related work in Sec. 2. Results for most methods are taken from the respective papers or BBN [56]. For the methods that did not report results on CIFAR-100-LT or mini-ImageNet-LT, we present additional details of our re-implementation.

- For CE and CosineCE, we choose hyperparameters (learning rate scheduler, number of epochs, etc.) to match the performance of CE on a previously reported work, BBN [56].
- *cRT [20] does not provide results on the CIFAR-100-LT and mini-ImageNet-LT datasets. We use their publicly available code to generate results for both the datasets.
Table 5: Comparison of our approach (TailCalib, TailCalibX) against previous works on mini-ImageNet-LT. Since we introduce using mini-ImageNet in a long-tail variant, all methods are our re-implementations. Best results are highlighted in bold, next best and notable results with underline.

| Type     | Method       | Many  | Mid   | Few   | All   |
|----------|--------------|-------|-------|-------|-------|
| Baseline | CE           | 57.77 | 29.42 | 8.06  | 32.94 |
|          | CosineCE     | 65.45 | 29.37 | 8.6   | 35.77 |
| Decouple | cRT [20]     | 66.00 | 31.99 | 29.3  | 43.09 |
| Distillation | CBD [17]   | 67.34 | 35.00 | 23.06 | 42.74 |
| Generation | MODALS [6]  | 66.62 | 28.65 | 11.06 | 36.67 |
|          | mixup [54]   | 64.71 | 27.51 | 8.1   | 34.72 |
| Ours     | CosineCE + TailCalib | 68.05 | 32.59 | 25.13 | 42.77 |
|          | CosineCE + TailCalibX | **68.3** | **34.3** | **29.4** | **44.73** |

- CBD is a recent, and to the best of our knowledge, unpublished work [17]. Through email conversations with the authors, we verified our implementation and present results with a single teacher trained on the both datasets.
- MODALS [6] was trained with the Gaussian noise augmentation. We choose \( \lambda = 0.01 \) after performing a sweep across \([0.1, 0.01, 0.001]\).
- For mixup [54], we choose \( \alpha = 0.01 \) after performing a sweep across \([0.01, 0.1, 0.2, 0.3, 0.5, 0.7]\).

CIFAR-100-LT (Table 4). Firstly, note that the CosineCE training paradigm is superior to CE, and in fact, outperforms several loss re-weighting based methods. CosineCE + TailCalibX outperforms all previous works ignoring distillation. In fact, it achieves performance close to CBD [17] (especially on imbalance factors 100 and 50) that trains two networks - a teacher and student, and benefits from the advantages of ensembling. Finally, applying TailCalibX on top of a trained CBD model results in further performance improvements of 1-2% notably for imbalance factors 100 and 50.

mini-ImageNet-LT (Table 5). TailCalibX outperforms all previous works. As compared to the CosineCE baseline, not only does it improve the accuracy for tail classes (few) by over 20%, we also see performance improvements for both the middle classes (mid, +4.93%) and head classes (many, +2.85%) as well. Surprisingly, CBD performs worse than classifier re-training (cRT), hence, we omit evaluation of TailCalibration as applied to a backbone trained with CBD.

4.5 Analysis

tSNE visualization. Fig. 4 shows a few feature embeddings computed using t-SNE [43] for head and tail categories from the CIFAR-100-LT dataset. For visual clarity, we show 10 validation samples for each class and up to 40 training + generated samples. We randomly choose 3 pairs of visually similar categories: tank and bus, woman and baby, and aquarium fish and turtle, such that one of them is from the bottom 15 tail categories, while the second belongs to the top 15 head categories.

In the left plot, before feature generation, we see that the validation samples (+) from the tail categories are often confused with visually similar and larger head classes: (i) five samples of the tank (red) are located close to the bus (blue) cluster; and (ii) roughly 7 of 10 samples of the woman category (purple) are potentially confused as belonging to the baby class (orange). In the right plot, we generate features using TailCalibration and re-compute the t-SNE embeddings. We observe a noticeable reduction in errors, even on this randomly picked toy sample. Now, 3 of 5 tank samples are close to the bus cluster, and 4 of 7 woman samples are close to the baby cluster, while the others are more easily separable.

Class nearest neighbors. Recall, to generate the calibrated tail distributions, we pick \( M \) categories as nearest neighbors for each feature \( z_{jk} \). We analyze the classes that are most commonly used as nearest neighbors for the bottom 15 tail classes of the CIFAR-100-LT dataset in Table 6. The table shows the nearest neighbors for imbalance factor 100 and \( M = 3 \). We observe that the class for which we wish to generate samples is often the nearest centroid and is ignored in the table for brevity.

This split of CIFAR-100-LT is sorted alphabetically, i.e. the classes with letter a belong to the head, and classes with letters towards the end of the alphabet form the tail. We see that nearest neighbors need not belong to the head classes, even though it may be common. For example, we see that tiger distribution draws from the lion class, or television from couch, possibly due to the fact that images with a television have additional furniture in them.
5 CONCLUSION AND FUTURE WORK

Learning to cope with long-tail distributions is an important problem for machine learning. Inspired by a recent work on few-shot learning [53], we explored one facet of this challenging task: feature generation as a means to balance the training data seen by a classifier. We analyzed the efficacy of distribution calibration and empirically validated that it can lead to substantial performance improvements. In this process, we achieved a new state-of-the-art on two synthetic datasets: CIFAR-100-LT and mini-ImageNet-LT, especially demonstrating that TailCalibration is orthogonal to, and can be combined with, other works such as CBD [17], to yield additional improvements.

Future work. In the future, we wish to analyze the feature generation process in a more thorough manner. In particular, we hope to discover the underlying class-specific manifolds on which the features would be expected to lie. While our initial attempts at evaluating TailCalibration on ImageNet did not yield significant improvement, we would like to extend our evaluation to other long-tail datasets, both synthetic: ImageNet-LT [26] and Places-LT [26], and natural: iNaturalist [44] or FineFoods dataset [32]. Another possibility is to exploit the hierarchy of categories and learn appropriate models such as hyperbolic representations [22] or their generalization to pseudo-Riemannian manifolds of constant nonzero curvature [23].

Acknowledgments. This work has been partly supported by the funding received from DST through the IMPRINT program (IMP / 2019 / 000250).

REFERENCES

[1] David Acuna, Guojun Zhang, Marc T. Law, and Sanja Fidler. 2021. f-Domain Adversarial Learning: Theory and Algorithms. In International Conference on Machine Learning (ICML)
[2] Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, Suvrit Sra, and Greg G Devroye. 2005. Clustering on the Unit Hypersphere using von Mises-Fisher Distributions. Journal of Machine Learning Research 6, 9 (2005).
[3] Kaide Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. 2019. Learning Imbalanced Datasets with Label-Distribution-Aware Margin Loss. In Advances in Neural Information Processing Systems (NeurIPS).
[4] Tianshuang Cao, Marc T Law, and Sanja Fidler. 2020. A Theoretical Analysis of the Number of Shots in Few-Shot Learning. In International Conference on Learning Representations (ICLR).
[5] Nitish V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research (JAIR) 16 (2002), 321–357.
[6] Tze-Him Cheung and Dit-Yan Yeung. 2021. MODALS: Modality-agnostic Automated Data Augmentation in the Latent Space. In International Conference on
Learning Representations (ICLR).

2020. Large Scale Visual Food Recognition.

2021. Large Scale Visual Representation Challenge.

2015. Large-Scale Long-Tailed Recognition in an Open World. In International Conference on Computer Vision (ICCV).

2020. Meta-weight-net: Learning an explicit mapping for sample weighting. In International Conference on Computer Vision (ICCV).

2020. Few-Shot Image Classification. In International Conference on Computer Vision (ICCV).

2015. Large-Scale Long-Tailed Recognition. In International Conference on Computer Vision (ICCV).

2019. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In International Conference on Computer Vision (ICCV).

2017. SGDR: Stochastic gradient descent with warm restarts. In International Conference on Learning Representations (ICLR).

2016. Deep Residual Learning for Deep Neural Networks. In Conference on Computer Vision and Pattern Recognition (CVPR).

2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In International Conference on Computer Vision and Pattern Recognition (CVPR).

2013. Internally Consistent Training of Object Classifiers. In Conference on Computer Vision and Pattern Recognition (CVPR).

2020. Deep Residual Learning for Deep Neural Networks. In Conference on Computer Vision and Pattern Recognition (CVPR).

2020. Feature Generation for Long-tail Classification ICVGIP'21, December 2021, Jodhpur, India