Curvature-Balanced Feature Manifold Learning for Long-Tailed Classification

Yanbiao Ma, Licheng Jiao, Fang Liu, Shuyuan Yang, Xu Liu and Lingling Li
Key Laboratory of Intelligent Perception and Image Understanding of the Ministry of Education, Xidian University
ymbamail@stu.xidian.edu.cn, lchjiao@mail.xidian.edu.cn

Abstract

To address the challenges of long-tailed classification, researchers have proposed several approaches to reduce model bias, most of which assume that classes with few samples are weak classes. However, recent studies have shown that tail classes are not always hard to learn, and model bias has been observed on sample-balanced datasets, suggesting the existence of other factors that affect model bias. In this work, we systematically propose a series of geometric measurements for perceptual manifolds in deep neural networks, and then explore the effect of the geometric characteristics of perceptual manifolds on classification difficulty and how learning shapes the geometric characteristics of perceptual manifolds. An unanticipated finding is that the correlation between the class accuracy and the separation degree of perceptual manifolds gradually decreases during training, while the negative correlation with the curvature gradually increases, implying that curvature imbalance leads to model bias. Therefore, we propose curvature regularization to facilitate the model to learn curvature-balanced and flatter perceptual manifolds. Evaluations on multiple long-tailed and non-long-tailed datasets show the excellent performance and exciting generality of our approach, especially in achieving significant performance improvements based on current state-of-the-art techniques. Our work opens up a geometric analysis perspective on model bias and reminds researchers to pay attention to model bias on non-long-tailed and even sample-balanced datasets. The code and model will be made public.

1. Introduction

The imbalance of sample numbers in the dataset gives rise to the challenge of long-tailed visual recognition. Most previous works assume that head classes are always easier to be learned than tail classes, e.g., class re-balancing [8, 14, 24, 34, 37, 46, 52], information augmentation [23, 31, 35, 38, 39, 44, 56, 64, 67], decoupled training [10, 16, 29, 30, 71, 76], and ensemble learning [20, 36, 57, 58, 61, 72, 77] have been proposed to improve the performance of tail classes. However, recent studies [3, 50] have shown that classification difficulty is not always correlated with the number of samples, e.g., the performance of some tail classes is even higher than that of the head classes. Also, [49] observes differences in model performance across classes on non-long-tailed data, and even on balanced data. Therefore, it is necessary to explore the impact of other inherent characteristics of the data on the classification difficulty, and then improve the overall performance by mitigating the model bias under multiple sample number distribution scenarios.

Focal loss [37] utilizes the DNN’s prediction confidence on instances to evaluate the instance-level difficulty. [50] argues that for long-tailed problems, determining class-level difficulty is more important than determining instance-level difficulty, and therefore defines classification difficulty by evaluating the accuracy of each class in real-time. However, both methods rely on the model output and still cannot...
not explain why the model performs well in some classes and poorly in others. Similar to the number of samples, we would like to propose a measure that relies solely on the data itself to model class-level difficulty, which helps to understand how deep neural networks learn from the data. The effective number of samples \([14]\) tries to characterize the diversity of features in each class, but it introduces hyperparameters and would not work in a sample-balanced dataset.

Most data distributions obey the manifold distribution law \([33, 54]\), i.e., samples of each class are distributed near a low-dimensional manifold in the high-dimensional space. The manifold consisting of features in the embedding space is called a perceptual manifold [11]. The classification task is equivalent to distinguishing each perceptual manifold, which has a series of geometric characteristics. We speculate that some geometric characteristics may affect the classification difficulty, and therefore conduct an in-depth study.

**Main contributions of our work are:** (1) We systematically propose a series of measures for the geometric characteristics of point cloud perceptual manifolds in deep neural networks (Sec 3). (2) The effect of learning on the separation degree (Sec 4.1) and curvature (Sec 4.2) of perceptual manifolds is explored. We find that the correlation between separation degree and class accuracy decreases with training, while the negative correlation between curvature and class accuracy increases with training (Sec 4.3), implying that existing methods can only mitigate the effect of separation degree among perceptual manifolds (3) Curvature regularization is proposed to facilitate the model to learn curvature-balanced and flatter feature manifolds, thus improving the overall performance (Sec 5). Our approach effectively reduces the model bias on multiple long-tailed (Fig 1) and non-long-tailed datasets (Fig 8), showing excellent performance (Sec 6).

2. Related Work (Appendix A)

3. The Geometry of Perceptual Manifold

In this section, we systematically propose a series of geometric measures for perceptual manifolds in deep neural networks, and all the pseudocode is in Appendix C.

3.1. Perceptual Manifold

A perceptual manifold is generated when neurons are stimulated by objects with different physical characteristics from the same class. Sampling along the different dimensions of the manifold corresponds to changes in specific physical characteristics. It has been shown \([33, 54]\) that the features extracted by deep neural networks obey the manifold distribution law. That is, features from the same class are distributed near a low-dimensional manifold in the high-dimensional feature space. Given data \(X = [x_1, \ldots, x_m]\) from the same class and a deep neural network \(Model = \{f(x, \theta_1), g(z, \theta_2)\}\), where \(f(x, \theta_1)\) represents a feature sub-network with parameters \(\theta_1\) and \(g(z, \theta_2)\) represents a classifier with parameters \(\theta_2\). Extract the \(p\)-dimensional features \(Z = [z_1, \ldots, z_m] \in \mathbb{R}^{p \times m}\) of \(X\) with the trained model, where \(z_i = f(x_i, \theta_1) \in \mathbb{R}^p\). Assuming that the features \(Z\) belong to class \(c\), the \(m\) features form a \(p\)-dimensional point cloud manifold \(M^c\), which is called a perceptual manifold [12].

3.2. The Volume of Perceptual Manifold

We measure the volume of the perceptual manifold \(M^c\) by calculating the size of the subspace spanned by the features \(z_1, \ldots, z_m\). First, the sample covariance matrix of \(Z\) can be estimated as \(\Sigma_Z = \mathbb{E}[\frac{1}{n} \sum_{i=1}^n z_i z_i^T] = \frac{1}{n} ZZ^T \in \mathbb{R}^{p \times p}\). Diagonalize the covariance matrix \(\Sigma_Z\) as \(UDU^T\), where \(D = diag(\lambda_1, \ldots, \lambda_p)\) and \(U = [u_1, \ldots, u_p] \in \mathbb{R}^{p \times p}\). \(\lambda_i\) and \(u_i\) denote the \(i\)-th eigenvalue of \(\Sigma_Z\) and its corresponding eigenvector, respectively. Let the singular value of matrix \(Z\) be \(\sigma_i = \sqrt{\lambda_i}\) for \(i = 1, \ldots, p\). According to the geometric meaning of singular value [1], the volume of the space spanned by vectors \(z_1, \ldots, z_m\) is proportional to the product of the singular values of matrix \(Z\), i.e., \(Vol(Z) \propto \prod_{i=1}^p \sigma_i = \prod_{i=1}^p \sqrt{\lambda_i}\). Considering \(\prod_{i=1}^p \lambda_i = det(\Sigma_Z)\), the volume of the perceived manifold is therefore denoted as \(Vol(Z) \propto \sqrt{det(\frac{1}{m} ZZ^T)}\).

However, when \(\frac{1}{m} ZZ^T\) is a non-full rank matrix, its determinant is 0. For example, the determinant of a planar point set located in three-dimensional space is 0 because its covariance matrix has zero eigenvalues, but obviously the volume of the subspace is not zero. We want to obtain the “area” of the planar point set, which is a generalized volume. We avoid the non-full rank case by adding the unit matrix \(I\) to the covariance matrix \(\frac{1}{m} ZZ^T\). \(I + \frac{1}{m} ZZ^T\) is a positive definite matrix with eigenvalues \(\lambda_i + 1(i = 1, \ldots, p)\). The above operation enables us to calculate the volume of a low-dimensional manifold embedded in high-dimensional space. The volume \(Vol(Z)\) of the perceptual manifold is proportional to \(\sqrt{det(I + \frac{1}{m} ZZ^T)}\). Considering the numerical stability, we further perform a logarithmic transformation on \(\sqrt{det(I + \frac{1}{m} ZZ^T)}\) and define the volume of the perceptual manifold as

\[
Vol(Z) = \frac{1}{2} \log_2 \det(I + \frac{1}{m} (Z - Z_{mean})(Z - Z_{mean})^T),
\]

where \(Z_{mean}\) is the mean of \(Z\). When \(m > 1\), \(Vol(Z) > 0\). Since \(I + \frac{1}{m} (Z - Z_{mean})(Z - Z_{mean})^T\) is a positive definite matrix, its determinant is greater than 0. In the following, the degree of separation between perceptual manifolds will be proposed based on the volume of perceptual manifolds.
3.3. The Separation Degree of Perceptual Manifold

Given the perceptual manifolds \( M^1 \) and \( M^2 \), they consist of point sets \( Z_1 = [z_{1,1}, \ldots, z_{1,m_1}] \in \mathbb{R}^{p \times m_1} \) and \( Z_2 = [z_{2,1}, \ldots, z_{2,m_2}] \in \mathbb{R}^{p \times m_2} \), respectively. The volumes of \( M^1 \) and \( M^2 \) are calculated as \( \text{Vol}(Z_1) \) and \( \text{Vol}(Z_2) \). Consider the following case, assuming that \( M^1 \) and \( M^2 \) have partially overlapped, when \( \text{Vol}(Z_1) \ll \text{Vol}(Z_2) \), it is obvious that the overlapped volume accounts for a larger proportion of the volume of \( M^1 \), when the class corresponding to \( M^1 \) is more likely to be confused. Therefore, it is necessary to construct an asymmetric measure for the degree of separation between multiple perceptual manifolds, and we expect this measure to accurately reflect the relative magnitude of the degree of separation.

Suppose there are \( C \) perceptual manifolds \( \{M^i\}_{i=1}^C \), which consist of point sets \( Z_i = [z_{i,1}, \ldots, z_{i,m_i}] \in \mathbb{R}^{p \times m_i} \) \( i = 1, \ldots, C \). Let \( Z = [Z_1, \ldots, Z_C] \subseteq \mathbb{R}^{p \times \sum_{j = 1}^{C} m_j} \), \( Z' = [Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_C] \subseteq \mathbb{R}^{p \times ((\sum_{j = 1}^{C} m_j) - m_i)} \), we define the degree of separation between the perceptual manifold \( M^i \) and the rest of the perceptual manifolds as

\[
S(M^i) = \frac{\text{Vol}(Z) - \text{Vol}(Z')}{\text{Vol}(Z_i)}.
\]

The following analysis is performed for the case when \( C = 2 \) and \( \text{Vol}(Z_2) > \text{Vol}(Z_1) \). According to our motivation, the measure of the degree of separation between perceptual manifolds should satisfy \( S(M^2) > S(M^1) \).

If \( S(M^2) > S(M^1) \) holds, then we can get

\[
\text{Vol}(Z) \text{Vol}(Z_1) - \text{Vol}(Z_i)^2 > \text{Vol}(Z) \text{Vol}(Z_2) - \text{Vol}(Z_2)^2,
\]

\[
\Leftrightarrow \text{Vol}(Z)(\text{Vol}(Z_1) - \text{Vol}(Z_2)) > \text{Vol}(Z_1)^2 - \text{Vol}(Z_2)^2,
\]

\[
\Leftrightarrow \text{Vol}(Z) < \text{Vol}(Z_1) + \text{Vol}(Z_2).
\]

We prove that \( \text{Vol}(Z) < \text{Vol}(Z_1) + \text{Vol}(Z_2) \) holds when \( \text{Vol}(Z_2) > \text{Vol}(Z_1) \) and the detailed proof is in Appendix B. The above analysis shows that the proposed measure meets our requirements and motivation. The formula for calculating the degree of separation between perceptual manifolds can be further reduced to

\[
S(M^i) = \log \delta \det \left( (I + \frac{Z'Z'^T}{\sum_{j = 1, j \neq i}^C m_j})^{-1}(I + \frac{ZZ^T}{\sum_{j = 1}^C m_j}) \right),
\]

\[
\delta = \det(I + \frac{1}{m} Z_i Z_i^T).
\]

The detailed derivation is in Appendix B. Next, we validate the proposed measure of the separation degree between perceptual manifolds in a 3D spherical point cloud scene. Specifically, we conducted the experiments in two cases:

1. Construct two 3D spherical point clouds of radius 1, and then increase the distance between their spherical centers. Since the volumes of the two spherical point clouds are equal, their separation degrees should be symmetric. The variation curves of the separation degrees are plotted in Fig 2, and it can be seen that the experimental results satisfy our theoretical predictions.

2. Change the distance between the centers of two spherical point clouds of radius 1 and radius 1.5. Observe their separation degrees, the separation degrees of these two spherical point clouds should be asymmetric. Fig 2 shows that their separation degrees increase as the distance between their centers increases. Also, the manifold with a larger radius has a greater separation degree, and this experimental result conforms to our analysis and motivation.

The separation degree between perceptual manifolds may affect the model’s bias towards classes. In addition, it can also be used as the regularization term of the loss function or applied in contrast learning to keep the different perceptual manifolds away from each other.

3.4. The Curvature of Perceptual Manifold

Given a point cloud perceptual manifold \( M \), which consists of a \( p \)-dimensional point set \( \{z_1, \ldots, z_n\} \), our goal is to calculate the Gauss curvature at each point. First, the normal vector at each point on \( M \) is estimated by the neighbor points. Denote by \( z_i^j \) the \( j \)-th neighbor point of \( z_i \) and \( u_i \) the normal vector at \( z_i \). We solve for the normal vector by minimizing the inner product of \( z_i^j - c_i, j = 1, \ldots, k \) and \( u_i [4] \), i.e.,

\[
\min \sum_{j=1}^{k} ((z_i^j - c_i)^Tu_i)^2,
\]

where \( c_i = \frac{1}{k} \sum_{j=1}^{k} z_i^j \) and \( k \) is the number of neighbor points. Let \( y_j = z_i^j - c_i \), then the optimization objective is converted to

\[
\min \sum_{j=1}^{k} (y_j^Tu_i)^2 = \min \sum_{j=1}^{k} u_i^Ty_jy_j^Tu_i = \min (u_i^T \sum_{j=1}^{k} y_jy_j^T)u_i).
\]

\( \sum_{j=1}^{k} y_jy_j^T \) is the covariance matrix of \( k \) neighbors of \( z_i \). Therefore, let \( Y = [y_1, \ldots, y_k] \in \mathbb{R}^{p \times k} \) and \( \sum_{j=1}^{k} y_jy_j^T = YY^T \). The optimization objective is further equated to

\[
\begin{aligned}
\begin{cases}
  f(u_i) = u_i^T YY^T u_i, YY^T \in \mathbb{R}^{p \times p}, \\
  \min(f(u_i)), \\
  s.t. u_i^T u_i = 1.
\end{cases}
\end{aligned}
\]
Construct the Lagrangian function \( L(u_i, \lambda) = f(u_i) - \lambda(u_i^T u_i - 1) \) for the above optimization objective, where \( \lambda \) is a parameter. The first-order partial derivatives of \( L(u_i, \lambda) \) with respect to \( u_i \) and \( \lambda \) are

\[
\frac{\partial L(u_i, \lambda)}{\partial u_i} = \frac{\partial}{\partial u_i} f(u_i) - \lambda \frac{\partial}{\partial u_i} (u_i^T u_i - 1) = 2(YY^T u_i - \lambda u_i),
\]

\[
\frac{\partial L(u_i, \lambda)}{\partial \lambda} = u_i^T u_i - 1.
\]

Let \( \frac{\partial L(u_i, \lambda)}{\partial u_i} \) and \( \frac{\partial L(u_i, \lambda)}{\partial \lambda} \) be 0, we can get \( YY^T u_i = \lambda u_i, u_i^T u_i = 1 \). It is obvious that solving for \( u_i \) is equivalent to calculating the eigenvectors of the covariance matrix \( YY^T \), but the eigenvectors are not unique. From \( \langle YY^T u_i, u_i \rangle = \langle \lambda u_i, u_i \rangle \) we can get \( \lambda = \langle YY^T u_i, u_i \rangle = u_i^T YY^T u_i \), so the optimization problem is equated to arg \( \min_{u_i} \lambda \). Performing the eigenvalue decomposition on the matrix \( YY^T \) yields \( p \) eigenvalues \( \lambda_1, \ldots, \lambda_p \) and the corresponding \( p \)-dimensional eigenvectors \( \xi_1, \ldots, \xi_p \) in \( \mathbb{R}^{p \times p} \), where \( \lambda_1 \geq \ldots \geq \lambda_p \geq 0, \|\xi_i\|_2 = 1, i = 1, \ldots, p, \langle \xi_i, \xi_i \rangle = 0 (a \neq b) \). The eigenvector \( \xi_{m+1} \) corresponding to the smallest non-zero eigenvalue of the matrix \( YY^T \) is taken as the normal vector \( u_i \) of \( M \) at \( z_i \).

Consider an \( m \)-dimensional affine space with center \( z_i \), which is spanned by \( \xi_1, \ldots, \xi_m \). This affine space approximates the tangent space at \( z_i \) on \( M \). We estimate the curvature of \( M \) at \( z_i \) by fitting a quadratic hypersurface in the tangent space utilizing the neighbor points of \( z_i \). The \( k \) neighbors of \( z_i \) are projected into the affine space \( z_i + \langle \xi_1, \ldots, \xi_m \rangle \) and denoted as

\[
o_j = [(z_i^j - z_i) \cdot \xi_1, \ldots, (z_i^j - z_i) \cdot \xi_m]^T \in \mathbb{R}^m, j = 1, \ldots, k.
\]

Denote by \( o_j[m] \) the \( m \)-th component \( (z_i^j - z_i) \cdot \xi_m \) of \( o_j \). We use \( z_i \) and \( k \) neighbor points to fit a quadratic hypersurface \( f(\theta) \) with parameter \( \theta \in \mathbb{R}^{m \times m} \). The hypersurface equation is denoted as

\[
f(o_j, \theta) = \frac{1}{2} \sum_{a,b} \theta_{a,b} o_j[a] o_j[b], j \in \{1, \ldots, k\},
\]

further, minimize the squared error

\[
E(\theta) = \sum_{j=1}^k \left( \frac{1}{2} \sum_{a,b} \theta_{a,b} [a] o_j[b] - (z_i^j - z_i) \cdot u_i \right)^2.
\]

Let \( \frac{\partial E(\theta)}{\partial \theta_{a,b}} = 0, a, b \in \{1, \ldots, m\} \) yield a nonlinear system of equations, but it needs to be solved iteratively. Here, we propose an ingenious method to fit the hypersurface and give the analytic solution of the parameter \( \theta \) directly. Expand the parameter \( \theta \) of the hypersurface into the column vector

\[
\theta = [\theta_{1,1}, \ldots, \theta_{1,m}, \theta_{2,1}, \ldots, \theta_{m,m}]^T \in \mathbb{R}^{m^2}.
\]
dataset, almost all samples lie on convex locations, and thus the complexity of the perceptual manifold is defined as the average $\frac{1}{n} \sum_{i=1}^{n} G(z_i)$ of the Gauss curvatures at all points on $M$. Our approach does not require iterative optimization and can be quickly deployed in a deep neural network to calculate the Gauss curvature of the perceptual manifold. Taking the two-dimensional surface in Fig 3 as an example, the surface complexity increases as the surface curvature is artificially increased. This indicates that our proposed complexity measure of perceptual manifold can accurately reflect the changing trend of the curvature degree of the manifold. In addition, Fig 3 shows that the selection of the number of neighboring points hardly affects the monotonicity of the complexity of the perceptual manifold. In our work, we select the number of neighboring points to be 40.

4. Learning How to Shape Perceptual Manifold

The perceptual manifolds in feature space are further decoded by the classification network into predicted probabilities for classification. Intuitively, we speculate that a perceptual manifold is easier to be decoded by the classification network when it is farther away from other perceptual manifolds and flatter. We provide more geometric views on classification and object detection in Appendix I. A model is usually considered to be biased when its performance on classes is inconsistent. In the following, we investigate the effect of the geometry of the perceptual manifold on the model bias and summarize three experimental discoveries.

4.1. Learning Facilitates The Separation

Learning typically leads to greater inter-class distance, which equates to greater separation between perceptual manifolds. We trained VGG-16 [48] and ResNet-18 [22] on F-MNIST [62] and CIFAR-10 [32] to explore the effect of the learning process on the separation degree between perceptual manifolds and observed the following phenomenon.

As shown in Fig 4, each perceptual manifold is gradually separated from the other manifolds during training. It is noteworthy that the separation is faster in the early stage of training, and the increment of separation degree gradually decreases in the later stage. Separation curves of perceptual manifolds for more classes are presented in Appendix D.

Figure 4. The variation curves between the separation degree of perceptual manifolds and training epochs on both datasets.

4.2. Learning Reduces The Curvature

Experiments are conducted with VGG-16 and ResNet-18 trained on F-MNIST and CIFAR-10, and we find that the perceptual manifold gradually flattens out during training. As shown in Fig 5, the curvature of the perceptual manifold decreases faster in the early stage of training, and it gradually becomes flat with further training. The curvature change curves of perceptual manifolds for more classes are shown in Appendix E.

4.3. Curvature Imbalance and Model Bias

Since learning separates perceptual manifolds from each other and also makes perceptual manifolds flatter, it is reasonable to speculate that the separation degree and curvature of perceptual manifolds correlate with class-wise classification difficulty. Experiments are conducted with VGG-16 and ResNet-18 trained on F-MNIST and CIFAR-10.

Figure 5. The variation curves between the complexity of perceptual manifolds and training epochs on both datasets.

Each class corresponds to a perceptual manifold. As shown in Fig 6, we observe that the negative correlation between the separation degree of the perceptual manifolds and the accuracy of the corresponding class decreases with training, while the correlation between the curvature and the accuracy increases. This implies that existing methods can only mitigate the effect of the separation degree between perceptual manifolds on the model bias, while ignoring the effect of perceptual manifold complexity on the model bias.

5. Curvature-Balanced Feature Learning

The above study shows that it is necessary to focus on the model bias caused by the curvature imbalance among perceptual manifolds. In this section, we propose curvature
regularization, which can reduce the model bias and further improve the performance of existing methods.

5.1. Design Principles of The Proposed Approach

The proposed curvature regularization needs to satisfy the following three principles to learn curvature-balanced and flat perceptual manifolds.

(1) The greater the curvature of a perceptual manifold, the stronger the penalty for it. Our experiments show that learning reduces the curvature, so it is reasonable to assume that flatter perceptual manifolds are easier to decode. (2) When the curvature is balanced, the penalty strength is the same for each perceptual manifold. (3) The sum of the curvatures of all perceptual manifolds tends to decrease.

5.2. Curvature Regularization (CR)

Given a C classification task, the p-dimensional feature embeddings of images from each class are represented as 

\[ Z_i = [z_{i1}, \ldots, z_{im}], i = 1, \ldots, C. \]

The mean Gaussian curvature \( G_i \), \( i = 1, \ldots, C \) of the corresponding perceptual manifold is calculated with the feature embeddings of each class (Appendix C: Algorithm 5). First, take the inverse of the curvature \( G_i \) and perform the maximum normalization on it. Then the negative logarithmic transformation is executed on the normalized curvature, and the curvature penalty term of the perceptual manifold \( M' \) is

\[ -\log(\frac{G_i^{-1}}{\max\{G_1^{-1}, \ldots, G_C^{-1}\}}). \]

Further, the overall curvature regularization term is denoted as

\[ L_{\text{Curvature}} = \sum_{i=1}^{C} -\log(\frac{G_i^{-1}}{\max\{G_1^{-1}, \ldots, G_C^{-1}\}}). \]

The detailed derivation is shown in Appendix F. In the following, we verify whether \( L_{\text{Curvature}} \) satisfies the three principles one by one.

(1) When the curvature \( G_i \) of the perceptual manifold is larger, \( G_i^{-1} \) is smaller. Since \( -\log(\cdot) \) is monotonically decreasing, \(-\log(\frac{G_i^{-1}}{\max\{G_1^{-1}, \ldots, G_C^{-1}\}})\) increases with \( G_i \) increases. \( L_{\text{Curvature}} \) is consistent with Principle 1.

(2) When \( G_1 = \cdots = G_C \), \( \max\{G_1^{-1}, \ldots, G_C^{-1}\} = G_1^{-1} = \cdots = G_C^{-1} \), so \(-\log(\frac{G_i^{-1}}{\max\{G_1^{-1}, \ldots, G_C^{-1}\}}) = 0, i = 1, \ldots, C \). \( L_{\text{Curvature}} \) follows Principle 2.

(3) The curvature penalty term of the perceptual manifold \( M' \) is 0 when \( G_i = \min\{G_1, \ldots, G_C\} \). Since the greater the curvature, the greater the penalty, our method aims to bring the curvature of all perceptual manifolds down to \( \min\{G_1, \ldots, G_C\} \). Obviously, \( \sum_{i=1}^{C} G_i \geq C \cdot \min\{G_1, \ldots, G_C\} \), so our approach promotes curvature balance while also making all perceptual manifolds flatter, which satisfies Principle 3.

The curvature regularization can be combined with any loss function. Since the correlation between curvature and accuracy increases with training, we balance the curvature regularization with other losses using a logarithmic function with a hyperparameter \( \tau \), and the overall loss is denoted as

\[ L = L_{\text{original}} + \frac{\log_{\text{epoch}}}{L_{\text{Curvature}}_{\text{original}}} \times L_{\text{Curvature}}, \tau > 1. \]

The term \( L_{\text{Curvature}}_{\text{original}}.\text{detach()} \) aims to make the curvature regularization loss of the same magnitude as the original loss. We investigate reasonable values of \( \tau \) in experiments (Sec 6.2). The design principle of curvature regularization is compatible with the learning objective of the model, and our experiments show that the effect of curvature imbalance on model bias has been neglected in the past. Thus curvature regularization is not in conflict with \( L_{\text{original}} \), as evidenced by our outstanding performance on multiple datasets.

5.3. Dynamic Curvature Regularization (DCR)

The curvature of perceptual manifolds varies with the model parameters during training, so it is necessary to update the curvature of each perceptual manifold in real-time. However, there is a challenge: only one batch of features is available at each iteration, and it is not possible to obtain all the features to calculate the curvature of the perceptual manifolds. If the features of all images from the training set are extracted using the current network at each iteration, it will greatly increase the time cost of training.

Algorithm 1 End-to-end training with DCR

| Require: |
| Training set \( D = \{(x_i, y_i)\}_{i=1}^M \), A CNN \( f(x, \theta_1), g(x, \theta_2) \), where \( f(\cdot) \) and \( g(\cdot) \) denote the feature sub-network and classifier, respectively. The training epoch is \( N \). |
| 1: Initialize the storage pool \( Q \) |
| 2: for epoch = 1 to \( N \) do |
| 3: \( \text{for iteration} = 0 \text{ to } \frac{M}{\text{batch size}} \) do |
| 4: Sample a mini-batch \( \{(x_i, y_i)\}_{i=1}^\text{batch size} \) from \( D \). |
| 5: Calculate feature embeddings \( z_i = f(x_i, \theta_1), i = 1, \ldots, \text{batch size} \). |
| 6: Store \( z_i \) and label \( y_i \) into \( Q \). |
| 7: if epoch < \( \nu \) then |
| 8: if epoch > 1 then |
| 9: Dequeue the oldest batch of features from \( Q \). |
| 10: end if |
| 11: Calculate loss \( L_{\text{original}} \). |
| 12: else |
| 13: Dequeue the oldest batch of features from \( Q \). |
| 14: Calculate the curvature of each perceptual manifold. |
| 15: Calculate loss: \( L = L_{\text{original}} + \frac{\log_{\text{epoch}}}{L_{\text{Curvature}}_{\text{original}}.\text{detach()}} \times L_{\text{Curvature}} \). |
| 16: end if |
| 17: Perform back propagation: \( L.\text{backward()} \). |
| 18: \( \text{optimizer.step()} \). |
| 19: end for |
| 20: end for |
Inspired by [3, 40], we design a first-in-first-out storage pool to store the latest historical features of all images. The slow drift phenomenon of features found by [59] ensures the reliability of using historical features to approximate the current features. We show the training process in Algorithm 1. Specifically, the features of all batches are stored in the storage pool at the first epoch. To ensure that the drift of the features is small enough, it is necessary to train another $n$ epochs to update the historical features. Experiments of [3] on large-scale datasets show that $n$ taken as 5 is sufficient, so $n$ is set to 5 in this work. When $e_{poch} > n$, the oldest batch of features in the storage pool is replaced with new features at each iteration, and the curvature of each perceptual manifold is calculated using all features in the storage pool. The curvature regularization term is updated based on the latest curvature. It should be noted that for decoupled training, CR is applied in the feature learning stage. Our method is employed in training only and does not affect the inference speed of the model.

6. Experiments

6.1. Datasets and Implementation Details

We comprehensively evaluate the effectiveness and generality of curvature regularization on both long-tailed and non-long-tailed datasets. The experiment is divided into two parts, the first part tests curvature regularization on four long-tailed datasets, namely CIFAR-10-LT, CIFAR-100-LT [14], ImageNet-LT [14, 47], and iNaturalist2018 [55]. The second part validates the curvature regularization on two non-long tail datasets, namely CIFAR-100 [32] and ImageNet [47]. For a fair comparison, the training and test images of all datasets are officially split, and the Top-1 accuracy on the test set is utilized as a performance metric. In addition, we train models on CIFAR-100, CIFAR-10/100-LT with a single NVIDIA 2080Ti GPU and ImageNet, ImageNet-LT, and iNaturalist2018 with eight NVIDIA 2080Ti GPUs. Please refer to Appendix G for a detailed description of the dataset and experimental setup.

6.2. Effect of $\tau$

When $\tau = e_{poch}$, $\log_{e} e_{poch} = 1$, so the selection of $\tau$ is related to the number of epochs. When the correlation between curvature and accuracy exceeds the correlation between the separation degree and accuracy, we expect $\log_{e} e_{poch} > 1$, which means that the curvature regularization loss is greater than the original loss. Following the [45] setting, all models are trained for 200 epochs, so $\tau$ is less than 200. To search for the proper value of $\tau$, experiments are conducted for CE + CR with a range of $\tau$, and the results are shown in Fig 7. Large-scale datasets require more training epochs to keep the perceptual manifolds away from each other, while small-scale datasets can achieve this faster, so we set $\tau = 100$ on CIFAR-10/100-LT and CIFAR-100, and $\tau = 120$ on ImageNet, ImageNet-LT, and iNaturalist2018.

6.3. Experiments on Long-Tailed Datasets

6.3.1 Evaluation on CIFAR-10/100-LT

Table 1 summarizes the improvements of CR for several state-of-the-art methods on long-tailed CIFAR-10 and CIFAR-100, and we observe that CR significantly improves all methods. For example, in the setting of IP 200, CR results in performance gains of 2.3%, 2.1%, and 1.5% for CE, Focal loss [37], and CB loss [14], respectively. When CR is applied to feature training, the performance of BBN [77] is improved by more than 1% on each dataset, which again validates that curvature imbalance negatively affects the learning of classifiers. When CR is applied to several state-of-the-art methods (e.g., RIDE + CMO [45] (2022) and GCL [34] (2022)), CR achieves higher classification accuracy with all IF settings.

Table 1. Comparison on CIFAR-10-LT and CIFAR-100-LT. The accuracy (%) of Top-1 is reported. The best and second-best results are shown in underlined bold and bold, respectively.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CIFAR-10-LT</th>
<th>CIFAR-100-LT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backbone Net</td>
<td>ResNet-32</td>
<td></td>
</tr>
<tr>
<td>imbalance factor</td>
<td>200 100 50 10</td>
<td>200 100 50 10</td>
</tr>
<tr>
<td>MiSLAS [76]</td>
<td>77.3 82.1 85.7 90.0</td>
<td>42.3 47.0 52.3 63.2</td>
</tr>
<tr>
<td>LDAM-DRW [8]</td>
<td>- 77.0 81.0 88.2</td>
<td>- 42.0 46.6 58.7</td>
</tr>
<tr>
<td>Cross Entropy</td>
<td>65.6 70.3 74.8 86.3</td>
<td>34.8 38.2 43.8 55.7</td>
</tr>
<tr>
<td>+ CR</td>
<td>67.9 72.6 76.2 89.5</td>
<td>36.9 40.5 45.1 57.4</td>
</tr>
<tr>
<td>Focal Loss [37]</td>
<td>65.2 70.3 76.7 86.6</td>
<td>35.6 38.4 44.3 55.7</td>
</tr>
<tr>
<td>+ CR</td>
<td>67.3 71.8 79.1 88.4</td>
<td>37.5 40.2 45.2 58.3</td>
</tr>
<tr>
<td>CB Loss [14]</td>
<td>68.8 74.5 79.2 87.4</td>
<td>36.2 39.6 45.3 57.9</td>
</tr>
<tr>
<td>+ CR</td>
<td>70.3 75.8 79.8 89.1</td>
<td>38.5 40.7 46.8 59.2</td>
</tr>
<tr>
<td>BBN [77]</td>
<td>- 79.8 82.1 88.3</td>
<td>- 42.5 47.0 59.1</td>
</tr>
<tr>
<td>+ CR [34]</td>
<td>- 81.2 83.5 89.4</td>
<td>- 43.7 48.1 60.0</td>
</tr>
<tr>
<td>De-c-TDE [53]</td>
<td>- 80.6 83.6 88.5</td>
<td>- 44.1 50.3 59.6</td>
</tr>
<tr>
<td>+ CR</td>
<td>- 81.8 84.5 89.9</td>
<td>- 45.7 51.4 60.3</td>
</tr>
<tr>
<td>RIDE (4*) [58]</td>
<td>- - - -</td>
<td>- 48.7 59.0 58.4</td>
</tr>
<tr>
<td>+ CR</td>
<td>- - - -</td>
<td>- 49.8 59.8 59.5</td>
</tr>
<tr>
<td>RIDE + CMO [45]</td>
<td>- - - -</td>
<td>- 50.0 53.0 60.2</td>
</tr>
<tr>
<td>+ CR</td>
<td>- - - -</td>
<td>- 50.7 54.3 61.4</td>
</tr>
<tr>
<td>GCL [34]</td>
<td>79.0 82.7 85.5</td>
<td>44.9 48.7 53.6</td>
</tr>
<tr>
<td>+ CR</td>
<td>79.9 83.5 86.8</td>
<td>45.6 49.8 55.1</td>
</tr>
</tbody>
</table>
Table 2. Top-1 accuracy (%) of ResNext-50 [63] on ImageNet-LT and Top-1 accuracy (%) of ResNet-50 [22] on iNaturalist2018 for classification. The best and the second-best results are shown in **bold** and **underline bold**, respectively.

<table>
<thead>
<tr>
<th>Methods</th>
<th>ImageNet-LT</th>
<th>iNaturalist2018</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ResNet-30</td>
<td>ResNet-30</td>
</tr>
<tr>
<td></td>
<td>H  M  T</td>
<td>Overall</td>
</tr>
<tr>
<td>CR</td>
<td><strong>65.9</strong></td>
<td>37.5 7.70 44.4</td>
</tr>
<tr>
<td>+ CR</td>
<td><strong>65.1</strong></td>
<td>40.7 19.5 47.3</td>
</tr>
<tr>
<td>Focal Loss</td>
<td>67.0 41.0 13.1 47.2</td>
<td>- - - 61.1</td>
</tr>
<tr>
<td>+ CR</td>
<td>67.3 43.2 22.5 49.6</td>
<td>- - - 62.3</td>
</tr>
<tr>
<td>BBN</td>
<td><strong>43.3</strong> 45.9 43.7 44.7</td>
<td><strong>49.4</strong> 70.8 65.3 66.3</td>
</tr>
<tr>
<td>+ CR</td>
<td><strong>45.2</strong> 46.8 <strong>44.5</strong> 46.2</td>
<td>50.6 71.5 66.8 67.6</td>
</tr>
<tr>
<td>LDAM</td>
<td>60.0 49.2 31.9 51.1</td>
<td>- - - 64.6</td>
</tr>
<tr>
<td>+ CR</td>
<td>60.8 50.3 33.6 52.4</td>
<td><strong>69.3</strong> 66.7 61.9 65.7</td>
</tr>
<tr>
<td>LADE</td>
<td><strong>62.3</strong> 49.3 31.2 51.9</td>
<td>- - - 69.7</td>
</tr>
<tr>
<td>+ CR</td>
<td><strong>62.5</strong> 50.1 33.7 53.0</td>
<td><strong>72.5</strong> 70.4 65.7 70.6</td>
</tr>
<tr>
<td>MJB</td>
<td>61.6 48.4 39.0 52.1</td>
<td>- - - 70.0</td>
</tr>
<tr>
<td>+ CR</td>
<td>62.8 49.2 40.4 53.4</td>
<td><strong>73.1</strong> <strong>70.3</strong> <strong>66.0</strong> 70.8</td>
</tr>
<tr>
<td>RIDE (4*) [58]</td>
<td><strong>67.8</strong> 53.4 46.2 56.6</td>
<td><strong>70.9</strong> <strong>72.4</strong> 73.1 72.6</td>
</tr>
<tr>
<td>+ CR</td>
<td><strong>68.5</strong> 54.2 38.8 57.8</td>
<td><strong>71.0</strong> <strong>73.8</strong> <strong>74.3</strong> 73.5</td>
</tr>
<tr>
<td>RIDE + CMO [45]</td>
<td><strong>66.4</strong> 54.9 35.8 56.2</td>
<td><strong>70.7</strong> <strong>72.6</strong> 73.4 72.8</td>
</tr>
<tr>
<td>+ CR</td>
<td><strong>67.3</strong> 54.6 38.4 57.4</td>
<td><strong>71.6</strong> <strong>73.7</strong> <strong>74.9</strong> <strong>73.8</strong></td>
</tr>
</tbody>
</table>

Table 3. Comparison on ImageNet and CIFAR-100.

<table>
<thead>
<tr>
<th>Methods</th>
<th>ImageNet</th>
<th>CIFAR-100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CE</td>
<td>CE + CR</td>
</tr>
<tr>
<td>VGG16</td>
<td>71.6</td>
<td>72.7</td>
</tr>
<tr>
<td>BN-Inception</td>
<td>73.5</td>
<td>74.3</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>70.1</td>
<td>71.3</td>
</tr>
<tr>
<td>ResNet-34</td>
<td>73.5</td>
<td>74.6</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>76.0</td>
<td>76.8</td>
</tr>
<tr>
<td>DenseNet-201 [28]</td>
<td>77.2</td>
<td>78.0</td>
</tr>
<tr>
<td>SE-ResNet-50 [25]</td>
<td>77.6</td>
<td>78.3</td>
</tr>
<tr>
<td>ResNetX-101 [63]</td>
<td>78.8</td>
<td>79.7</td>
</tr>
</tbody>
</table>

6.4. Experiments on Non-Long-Tailed Datasets

Curvature imbalance may still exist on sample-balanced datasets, so we evaluate CR on non-long-tailed datasets. Table 3 summarizes the improvements of CR on CIFAR-100 and ImageNet for various backbone networks, and we observe that CR results in approximately 1% performance improvement for all backbone networks. In particular, the accuracy of CE + CR exceeds CE by 1.5% on CIFAR-100 when using ResNet-18 [22] as the backbone network. The experimental results show that our proposed curvature regularization is applicable to non-long-tailed datasets and compatible with existing backbone networks and methods.

6.5. Curvature Regularization Reduces Model Bias

Here we explore how curvature regularization improves the model performance. The measuring model bias with the variance of the accuracy of all classes [50], Fig 1 and Fig 8 show that curvature regularization reduces the bias of the models trained on CIFAR-100-LT, Image-Net-LT, CIFAR-100, and ImageNet. By combining Tables 1 and 2, it can be found that curvature regularization reduces the model bias mainly by improving the performance of the tail class and does not compromise the performance of the head class, thus improving the overall performance. In addition, In Appendix H we answer the following two questions: (1) Is the curvature more balanced after training with CR? (2) Did the correlation between curvature imbalance and class accuracy decrease after training with CR?

Figure 8. Curvature regularization reduces the bias of multiple backbone networks trained on ImageNet and CIFAR-100.

7. Conclusion

This work mines and explains the impact of data on the model bias from a geometric perspective, introducing the imbalance problem to non-long-tailed data and providing a geometric analysis perspective to drive toward fairer AI.
References


[24] Youngkyu Hong, Seungju Han, Kwanghee Choi, Seokjun Seo, Beomsu Kim, and Buru Chang. Disentangling label distribution for long-tailed visual recognition. In Proceedings of the IEEE/CVF conference on...
computer vision and pattern recognition, pages 6626–6636, 2021. 1, 8


[65] Yuzhe Yang and Zhi Xu. Rethinking the value of labels for improving class-imbalanced learning.


