Appendix A: Related Work

In practice, the dataset usually tends to follow a long-tailed distribution, which leads to models with very large variances in performance on each class. It should be noted that most researchers default to the main motivation for long-tail visual recognition is that classes with few samples are always weak classes. Therefore, numerous methods have been proposed to improve the performance of the model on tail classes. [73] divides these methods into three fields, namely class rebalancing, information augmentation, and module improvement. Unlike the above, [50] and [3] observe that the number of samples in the class does not exactly show a positive correlation with the accuracy, and the accuracy of some tail classes is even higher than the accuracy of the head class. Therefore, they propose to use other measures to gauge the learning difficulty of the classes rather than relying on the sample number alone. In the following, we first present past research up to [50] [3] and lead to our work.

Class Rebalancing

The motivation for class rebalancing is intuitive; researchers have argued that tail classes with fewer samples lead to an imbalance in class-level loss and thus an inconsistent degree of learning for each class. Therefore cost-sensitive learning [18, 66, 75, 78] and resampling [9, 19, 57, 74] are proposed to rebalance the losses.

Cost-sensitive learning is proposed to balance the losses incurred by all classes, usually by applying a larger penalty to the tail classes on the objective function (or loss function) [14, 37, 46, 49, 50, 52]. [46] proposes to adjust the loss with the label frequencies to alleviate class bias. [37] not only assigns weights to the loss of each class, but also assigns higher weights to hard samples. Recent studies have shown that the effect of reweighting losses by the inverse of the number of samples is modest [41, 42]. Some methods that produce more "smooth" weights for reweighting perform better, such as taking the square root of the number of samples as the weight [41]. [14] attributes the better performance of this smoother method to the existence of marginal effects. In addition, [2]proposes to learn the classifier with class-balanced loss by adjusting the weight decay and MaxNorm in the second stage.

Resampling methods are divided into oversampling and undersampling [19, 21, 30]. The idea of oversampling is to randomly sample the tail classes to equalize the number of samples and thus optimize the classification boundaries. The undersampling methods balance the number of samples by randomly removing samples from the head classes. For example, [58] finds that training with a balanced subset of a long-tailed dataset is instead better than using the full dataset. In addition, [30, 77] fine-tune the classifier via a resampling strategy in the second phase of decoupled training. [60] continuously adjusts the distribution of resampled samples and the weights of the two-loss terms during training to make the model perform better. [69] employs the model classification loss from an additional balanced validation set to adjust the sampling rate of different classes.

Information Augmentation

Class rebalancing is inherently unable to handle missing information because no additional information is introduced. Information augmentation aims to improve the performance on tail classes by introducing additional information into the model training. This method is classified into two types: knowledge transfer and data augmentation.

There are four main schemes of knowledge transfer, which are head-to-tail knowledge transfer, model pre-training, knowledge distillation, and self-training. Head-to-tail knowledge transfer aims to transfer knowledge from the head classes to the tail classes to improve the performance of the tail classes. FTL [67] assumes that the feature distributions of the common and UR classes (i.e., rare classes) have the same variance, so the variance from the head classes is used to guide the feature enhancement of the tail classes. LEAP [39] transfers the intra-class angle distribution of features to the tail classes and constructs a "feature cloud" centered on each feature to expand the distribution of the tail classes. Similar to the adversarial attack, M2m [31] proposes to transform some samples from the head class into the tail samples by perturbation-based optimization to achieve tail augmentation. OFA [10] decomposes the features of each class into class-generic features and class-specific features. During training, the tail class-specific features are fused with the head class-generic features to generate new features to augment the tail classes. GIST [38] proposes to transfer the geometric information of the feature distribution boundaries of the head classes to the tail classes by increasing the classifier weights of the tail classes. The motivation of the recently proposed CMO [45] is very intuitive, it argues that the images from the head classes have rich backgrounds, so the images from the tail classes can be pasted directly onto the rich background images of the head classes to increase the richness of the tail images. The remaining three types of schemes are relatively few. [15] first pre-trains the model with all the long-tailed samples, and then fine-tunes the model on a balanced training subset. [65] proposes to pre-train the model with self-supervised learning and perform standard training on the long-tailed data. LST [26] utilizes knowledge distillation to overcome catastrophic forgetting in incremental learning.

Data augmentation in long-tailed recognition improves the performance of tail classes by improving conventional data augmentation methods. MiSLAS [76] suggests adopting mixup to augment feature learning, while not using mixup in classifier learning. FASA [69] proposes to

generate features based on Gaussian prior and evaluate weak classes on a balanced dataset to adjust the sampling rate. MetaSAug [35] generates augmented features for tail classes with ISDA.

Module Improvement

In addition to information enhancement to improve performance from a data perspective, researchers have designed numerous network modules for long-tailed recognition. The methods in this section can be divided into representation learning, classifier design, decoupled training, and ensemble learning. Decoupled training divides the training process into representation learning and classifier learning. LMLE [27], CRL [17], KCL [29] and PaCo [13] introduce metric learning methods to increase the differentiation of the representation and make the model more robust to data distribution shifts. HFL [43] proposes to hierarchically cluster all classes into leaves of a tree and then improve the generalization performance of the tail classes by sharing the parameters of the parent nodes or similar leaves.

Ensemble learning has shown great potential in long-tailed recognition. BBN [77] designed a two-branch network to rebalance the classifier, which is consistent with the idea of decoupled training. To avoid decoupled training damaging the performance of the head class, SimCal [57] trained networks with dual branches, one for rebalancing the classifier and the other for maintaining the performance of the head class. ACE [7], RIDE [58], and TADE [72] introduced multiple experts with specific complementary capabilities, which led to a significant improvement in the overall performance of the model.

Class-Difficulty Based Methods

The study of class difficulty is most relevant to our work. The methods in the three domains presented above almost all assume that classes with few samples are the most difficult classes to be learned, and therefore more attention is given to these classes. However, recent studies [3, 50] have observed that the performance of some tail classes is even higher than that of the head classes, and that the performance of different classes varies on datasets with perfectly balanced samples. These phenomena suggest that the sample number is not the only factor that affects the performance of classes. The imbalance in class performance is referred to as the "bias" of the model, and [50] defines the model bias as

$$bias = \max(\frac{\max_{c=1}^{N} A_c}{\min_{c'=1}^{N} A_{c'} + \varepsilon} - 1, 0),$$

where A_c denotes the accuracy of the c-th class. When the accuracy of each class is identical, bias = 0. [50] computes the difficulty of class c using $1 - A_c$ and calculates the weights of the loss function using a nonlinear function of class difficulty. Unlike [50], [3] proposes a model-independent measure of classification difficulty, which directly utilizes the data matrix to calculate the semantic scale of each class to represent the classification difficulty. As with the sample number, model-independent measures can help us understand how deep neural networks learn from data. When we get data from any domain, if we can measure the difficulty of each class directly from the data, we can guide the researchers to collect the difficult classes in a targeted manner instead of blindly, greatly facilitating the efficiency of applying AI in practice.

In this work, we propose to consider the classification task as the classification of perceptual manifolds. The influence of the geometric characteristics of the perceptual manifold on the classification difficulty is further analyzed, and feature learning with curvature balanced is proposed.

Appendix B: The Proof and Derivation of Section 3.3

First, recall the definition of the degree of separation of the perceptual manifold as follows.

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

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

The following analysis is performed for the case when $C = 2$ and $Vol(Z_2) > Vol(Z_1)$. 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

$$\begin{aligned} &Vol(Z)Vol(Z_1) - Vol(Z_1)^2 > Vol(Z)Vol(Z_2) - Vol(Z_2)^2, \\ \iff\ &Vol(Z)(Vol(Z_1) - Vol(Z_2)) > Vol(Z_1)^2 - Vol(Z_2)^2, \\ \iff\ &Vol(Z) < Vol(Z_1) + Vol(Z_2). \end{aligned}$$

We prove that $Vol(Z) < Vol(Z_1) + Vol(Z_2)$ holds when $Vol(Z_2) > Vol(Z_1)$, and the details are as follows.

Proof *Since the function $\log_2 \det(\cdot)$ is strictly concave, the real symmetric positive definite matrices $I + \frac{1}{m} Z^T Z$ and $I + \frac{1}{m} diag\{Z_1^T Z_1, Z_2^T Z_2\}$ satisfy [6]*

$$\begin{aligned} \log_2 \det(I + \frac{1}{m} Z^T Z) \le \log_2 \det(I + \frac{1}{m} diag\{Z_1^T Z_1, Z_2^T Z_2\}) \\ + tr((I + \frac{1}{m} diag\{Z_1^T Z_1, Z_2^T Z_2\})^T (I + \frac{1}{m} Z^T Z)). \end{aligned}$$

Also because

$$\log_2 \det(I + \frac{1}{m} diag\{Z_1^T Z_1, Z_2^T Z_2\}) = \log_2 \det(I + \frac{1}{m} Z_1^T Z_1) + \log_2 \det(I + \frac{1}{m} Z_2^T Z_2)$$

and

$$tr((I + \frac{1}{m} diag\{Z_1^T Z_1, Z_2^T Z_2\})^T (I + \frac{1}{m} Z^T Z)) = tr(diag\{I, I\}) = m.$$

We can get

$$\log_2 \det(I + \frac{1}{m} Z^T Z) \leq \log_2 \det(I + \frac{1}{m} Z_1^T Z_1) + \log_2 \det(I + \frac{1}{m} Z_2^T Z_2),$$

i.e., $Vol(Z) < Vol(Z_1) + Vol(Z_2)$ *holds.*

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

$$\begin{aligned} S(M^i) &= \frac{\log_2 \det(I + \frac{1}{\sum_{j=1}^{C} m_j} ZZ^T)}{\log_2 \det(I + \frac{1}{m_i} Z_i Z_i^T)} - \frac{\log_2 \det(I + \frac{1}{\sum_{j=1}^{C} m_j} Z'Z'^T)}{\log_2 \det(I + \frac{1}{m_i} Z_i Z_i^T)} \\ &= \frac{\log_2 \frac{\det(I + \frac{1}{\sum_{j=1}^{C} m_j} ZZ^T)}{\det(I + \frac{1}{\sum_{j=1, j\neq i}^{C} m_j} Z'Z'^T)}}{\log_2 \det(I + \frac{1}{m_i} Z_i Z_i^T)} \\ &= \log_\delta \det((I + \frac{Z'Z'^T}{\sum_{j=1, j\neq i}^{C} m_j})^{-1} (I + \frac{ZZ^T}{\sum_{j=1}^{C} m_j})), \\ \delta &= \det(I + \frac{1}{m} Z_i Z_i^T). \end{aligned}$$

Appendix C: Pseudocode for Measure The Geometry Properties of Perceptual Manifold

In Section 3 we propose measures for the perceptual manifold's volume, separation, and curvature. We provide the pseudo-code below to show how to apply these methods in practice. Our approach can be applied not only to calculate the geometric properties of feature manifolds but also to the image space. In addition to image data, other types of data also obey the manifold distribution law, so our method can be employed to evaluate them as well.

Pseudocode for The Volume of Perceptual Manifold

We give the calculation procedure of perceptual manifold volume in Algorithm 2. In addition, the volume of the data manifold can also be calculated directly in image space (Algorithm 3). Unlike the method for calculating the volume of the perceptual manifold, it is necessary to shrink and flatten the image into vectors. The reason for this is that the dimensionality of the image space is often very high, so we alleviate the dimensionality catastrophe by simply downsampling the dimensions.

Pseudocode for The Separation Degree of Perceptual Manifold

The distributions of different classes are far from each other to give the model higher discriminative power. Euclidean distance or cosine distance between class centers is often used as the measure of distance between classes, and these two distances are also commonly used as loss functions when constructing sample pairs. However, maximizing the distance between proxy points or samples does not keep one class away from all the remaining classes at the same time, and the distance between class centers does not reflect the degree of overlap of distributions. **Therefore, we**

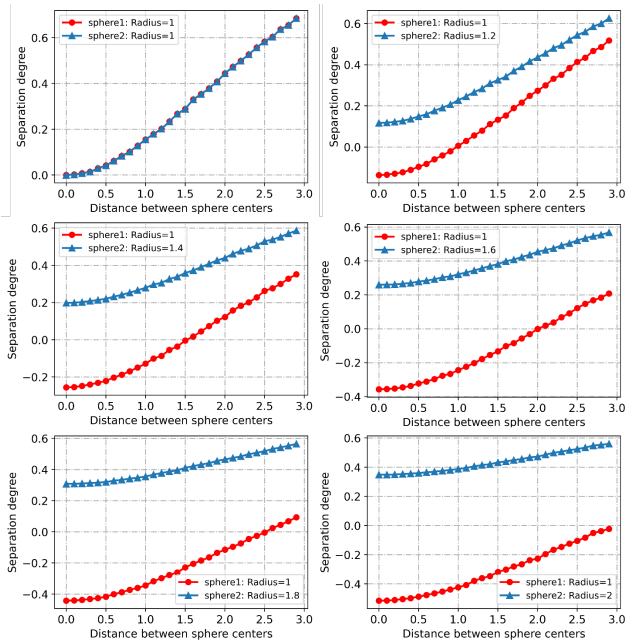

Figure 9. The curve of the degree of separation of two spherical point cloud that varies with the distance between spherical centers.

define the degree of separation of the perceptual manifold based on the volume of the perceptual manifold, which is applicable to the case of multiple perceptual manifolds and is an asymmetric measure. Algorithm 4 shows in detail how to calculate the degree of separation of the perceptual manifold in practice. In addition, we add more

Algorithm 2 Pseudocode for The Volume of Perceptual Manifold

Input: Training set $D = \{(x_i, y_i)\}_{i=1}^{M}$ with the total number C of classes. A CNN $\{f(x, \theta_1), g(z, \theta_2)\}$, where $f(\cdot)$ and $g(\cdot)$ denote the feature sub-network and classifier, respectively.
Output: The volume of all perceptual manifolds.

for $j = 1$ to C **do**
  Select the sample set $D_j = \{(x_i, y_i)\}_{i=1}^{m_j}$ for class j from D, m_j is the number of samples for class j.
  Calculate the feature embedding $Z_j = \{z_i \mid z_i = f(x_i, \theta_1)\}_{i=1}^{m_j}$ of D_j, $Z_j = \left[z_1, z_2, \ldots, z_{m_j}\right] \in \mathbb{R}^{p \times m_j}$.
  $Z_j = Z_j - NumPy.mean\left(Z_j, 1\right)$.
  Calculate the covariance matrix $\Sigma_j = \frac{1}{m_j} Z_j Z_j^T$.
  Calculate the volume $Vol\left(\Sigma_j\right) = \frac{1}{2}\log_2 \det\left(I + \Sigma_j\right)$ of the perceptual manifold corresponding to class j.
end for

Algorithm 3 Pseudocode for The Volume of Data Manifold

Input: Training set $D = \{(x_i, y_i)\}_{i=1}^{M}$ with the total number C of classes. A CNN $\{f(x, \theta_1), g(z, \theta_2)\}$, where $f(\cdot)$ and $g(\cdot)$ denote the feature sub-network and classifier, respectively.
Output: The volume of all data manifolds.

for $j = 1$ to C **do**
  Select the sample set $D_j = \{(x_i, y_i)\}_{i=1}^{m_j}$ for class j from D, m_j is the number of samples for class j.
  Resize the image to $(imagesize, imagesize, 3)$.
  Flatten the image into a vector of length $d = imagesize \times imagesize \times 3$ and store it in $D'_j = \left[x'_1, x'_2, \ldots, x'_{m_j}\right] \in \mathbb{R}^{d \times m_j}$.
  $D'_j = D'_j - NumPy.mean\left(D'_j, 1\right)$.
  Calculate the covariance matrix $\Sigma_j = \frac{1}{m_j} D'_j D'^T_j$.
  Calculate the sample volume $Vol\left(\Sigma_j\right) = \frac{1}{2}\log_2 \det\left(I + \Sigma_j\right)$ for class j.
end for

results to Fig 2 in Fig 9, where it can be clearly observed that as the difference in volume between the two spherical point cloud manifolds becomes larger, the difference in the degree of separation between the two increases, a result that is fully consistent with our motivation.

Pseudocode for the Mean Gaussian Curvature of The Perceptual Manifold

The complexity of the perceptual manifold reflects the extraction ability of the deep neural network for the input image [12]. [33] proposed to decompose the manifold into multiple pieces, each of which is homogeneously mapped to a linear space, and the lower limit of the number of pieces in all decomposition methods defines the complexity of the manifold. However, there is no way to know the quantitative expression of the point cloud perceptual manifold, and it is difficult to find its homogeneous mapping.

We give the analytical solution for estimating the curvature of the perceptual manifold in Section 3.4, but the whole derivation process is complicated. Therefore, we clearly list the steps for calculating the curvature of the perceptual manifold in Algorithm 5, so that it can be easily understood and used by other researchers.

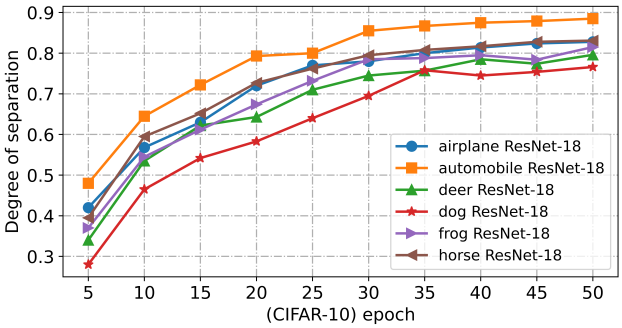

Figure 10. Curves of the separation degree of perceptual manifolds with training epoch on CIFAR-10.

Appendix D: More Experimental Results and Analysis for Section 4.1

We observed on Fashion-MNIST and CIFAR-10 that learning led to a progressive increase in the separation degree of the perceptual manifolds, and more results are added in Fig 10. It can be seen that the separation degree of all perceptual manifolds in CIFAR-10 increases with the training epoch. Section 4.3 presents the Pearson correlation coefficient between the separation degree of the perceptual manifold and the class accuracy over 0.6 in the early stages of

Algorithm 4 Pseudocode for The Separation Degree of Perceptual Manifold

Input: Training set $D = \{(x_i, y_i)\}_{i=1}^{M}$ with the total number C of classes. A CNN $\{f(x, \theta_1), g(z, \theta_2)\}$, where $f(\cdot)$ and $g(\cdot)$ denote the feature sub-network and classifier, respectively.
Output: The volume of all data manifolds.

1: **for** $j = 1$ to C **do**
2: Select the sample set $D_j = \{(x_i, y_i)\}_{i=1}^{m_j}$ for class j from D, m_j is the number of samples for class j.
3: Calculate the feature embedding $Z_j = \{z_i \mid z_i = f(x_i, \theta_1)\}_{i=1}^{m_j}$ of D_j, $Z_j = [z_1, z_2, \ldots, z_{m_j}] \in \mathbb{R}^{p \times m_j}$.
4: **end for**
5: There exist C perceptual manifolds $\{M^i\}_{i=1}^{C}$, which consist of point sets $\{Z_i = [z_{i,1}, \ldots, z_{i,m_i}] \in \mathbb{R}^{p \times m_i}\}_{i=1}^{C}$. Let $Z = [Z_1, \ldots, Z_C] \in \mathbb{R}^{p \times \sum_{j=1}^{C} m_j}$.
6: **for** $i = 1$ to C **do**
7: Let $Z' = [Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_C] \in \mathbb{R}^{p \times ((\sum_{j=1}^{C} m_j) - m_i)}$.
8: Calculate the degree of separation $S(M^i) = \log_\delta \det((I + \frac{Z'Z'^T}{\sum_{j=1, j \neq i}^{C} m_j})^{-1}(I + \frac{ZZ^T}{\sum_{j=1}^{C} m_j})), \delta = \det(I + \frac{1}{m} Z_i Z_i^T)$ for perceptual manifold M^i.
9: **end for**

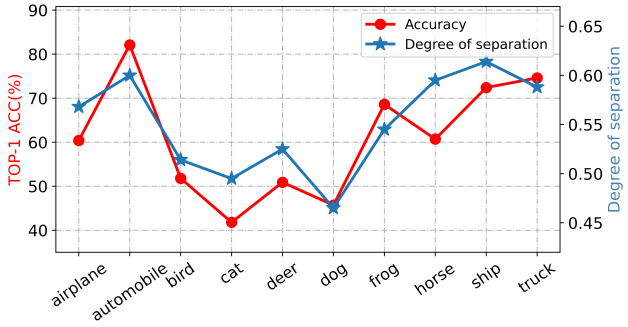

Figure 11. The degree of separation and corresponding class accuracy of all perceptual manifolds on CIFAR-10.

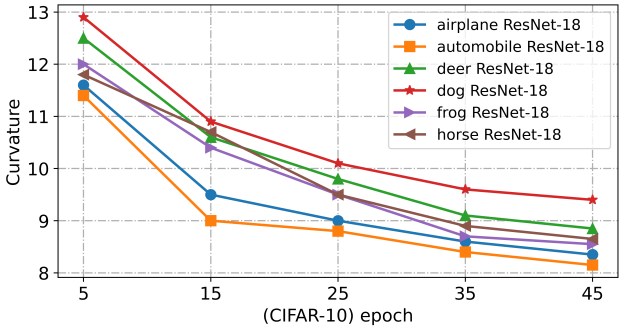

Figure 12. Curves of the complexity of perceptual manifolds with training epochs on CIFAR-10.

training, which we further validate. ResNet-18 was trained on CIFAR-10, and when the epoch reached 10, features of samples from all classes were extracted with ResNet-18, and the accuracy of each class was tested. The separation degree of each perceptual manifold is calculated utilizing the features, and then the separation degree of all perceptual manifolds and the corresponding class accuracy are plotted in Fig 11. We found that the two were indeed highly correlated, and additional experimental results provide a more detailed analysis for the discovery shown in Fig 6.

Appendix E: More Experimental Results and Analysis for Section 4.2

We add more results in Fig 12. The experiments amply show that learning makes each perceptual manifold flatter, which confirms our speculation that the flatter the perceptual manifold is, the easier it is to decode. It is important to note that the curvature of the different perceptual manifolds shows differences as the training epoch increases. The correlation between curvature of different magnitudes and class accuracy increases with increasing training epochs. We train ResNet-18 on CIFAR-10 and use ResNet-18 to extract features from all samples when the training epoch is 50. The inverse of the curvature of each perceptual manifold is calculated, while the accuracy of each class is tested, and both are plotted in Fig 13. It can be seen that the inverse of the curvature of perceptual manifold does have a high correlation with the corresponding class accuracy in the late training phase.

Appendix F: The Derivation of Curvature Regularization (Section 5.2)

Here, we will derive the final result step by step according to three design principles of curvature regularization. First recall the three principles.

(1) The greater the curvature of a perceptual manifold, the stronger the penalty for it.

(2) When the curvature is balanced, the penalty strength is the same for each perceptual manifold.

Algorithm 5 Pseudocode for the Mean Gaussian Curvature of The Perceptual Manifold

Input: Given a point cloud perceptual manifold M, which consists of a p-dimensional point set $\{z_1, \ldots, z_n\}$. Denote by z_i^j the j-th neighbor point of z_i and u_i the normal vector at z_i.
Output: The mean Gaussian curvature of the perceptual manifold M.

for $i = 1$ to n **do**
  Select k neighbor points $z_i^j, j = 1, \ldots, k$ of z_i and let $Y = [z_i, z_i^1, \ldots, z_i^k] \in \mathbb{R}^{p \times k}$.
  $Y = Y - NumPy.mean\,(Y, 1)$.
  Calculate the local covariance matrix $\frac{1}{k} Y Y^T$.
  Diagonalize $\frac{1}{k} Y Y^T$ as $U^T D U$ with $D = diag(\lambda_1, \ldots, \lambda_p), \lambda_1 \geq \cdots \geq \lambda_{m+1} > \lambda_{m+2} = \cdots = 0, U = [\xi_1, \ldots, \xi_p] \in \mathbb{R}^{p \times p}, \|\xi_i\|_2 = 1, i = 1, \ldots, p, \langle \xi_a, \xi_b \rangle = 0 (a \neq b)$.
  Let $u_i = \lambda_{m+1}$.
  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 $f(o_j, \theta) = \frac{1}{2} \sum_{a,b} \theta_{a,b} o_j[a]\, o_j[b], j \in \{1, \ldots, k\}$.
  Expand the parameter θ of the hypersurface into the column vector $\theta = [\theta_{1,1}, \ldots, \theta_{1,m}, \theta_{2,1}, \ldots, \theta_{m,m}]^T \in \mathbb{R}^{m^2}$..
  Organize the k neighbor points $\{o_j\}_{j=1}^k$ of z_i according to the following form: Organize the k neighbor points $\{o_j\}_{j=1}^k$ of z_i according to the following form:

$$O(z_i) = \begin{bmatrix} o_1[1]\, o_1[1] & o_1[1]\, o_1[2] & \cdots & o_1[m]\, o_1[m] \\ o_2[1]\, o_2[1] & o_2[1]\, o_2[2] & \cdots & o_2[m]\, o_2[m] \\ \vdots & \vdots & \ddots & \vdots \\ o_k[1]\, o_k[1] & o_k[1]\, o_k[2] & \cdots & o_k[m]\, o_k[m] \end{bmatrix} \in \mathbb{R}^{k \times m^2}.$$

  The target value is $T = \left[(z_i^1 - z_i) \cdot u_i, (z_i^2 - z_i) \cdot u_i, \ldots, (z_i^k - z_i) \cdot u_i\right]^T \in \mathbb{R}^k$.
  Solve for $\frac{\partial}{\partial \theta} (\frac{1}{2} tr \left[(O(z_i)\theta - T)^T (O(z_i)\theta - T)\right]) = 0$ to get $\theta = (O(z_i)^T O(z_i))^{-1} O(z_i)^T T$.
  The Gauss curvature of the perceptual manifold M at z_i can be calculated as $G(z_i) = det(\theta) = det((O(z_i)^T O(z_i))^{-1} O(z_i)^T T)$.
end for
The average Gaussian curvature $\frac{1}{n} \sum_{i=1}^n G(z_i)$ of the perceptual manifold M is the average of the Gauss curvatures at all points on M.

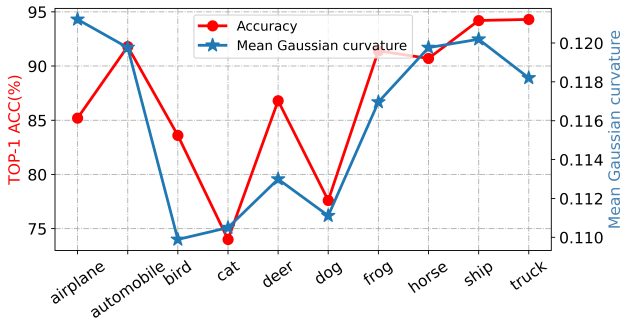

Figure 13. The inverse of the mean Gaussian curvature of all perceptual manifolds and the accuracy of all classes on CIFAR-10.

(3) The sum of the curvatures of all perceptual manifolds tends to decrease.

In order to propose curvature regularization in a reasonable way, we start from softmax cross-entropy loss to inspire our method. Given a C classification task, suppose a sample x is labeled as Y_k and it is predicted as each class with probabilities $P_1, P_2, \ldots, P_C$, respectively. The cross-entropy loss generated by sample x is calculated as $L(x) = \sum_{i=1}^C - Y_i \log(P_i)$, where $Y_k = 1, Y_i = 0, i \neq k$. The goal of $L(x)$ is to make $\log(P_k)$ converge to 0, i.e., P_k converges to 1, at which point $P_i (i \neq k)$ converges to 0. Unlike cross-entropy loss, which can pull apart the difference between P_k and other probabilities, we expect the mean Gaussian curvature of the C perceptual manifolds to converge to equilibrium.

Assume that the mean Gaussian curvatures of the C perceptual manifolds are $G_1, G_2, \ldots, G_C$, and perform the maximum normalization on them. The $-\log(G_k)$ loss can make G_k converge to 1. Therefore, perform a negative logarithmic transformation on the curvature of all perceived manifolds and use it as loss, which can make each curvature

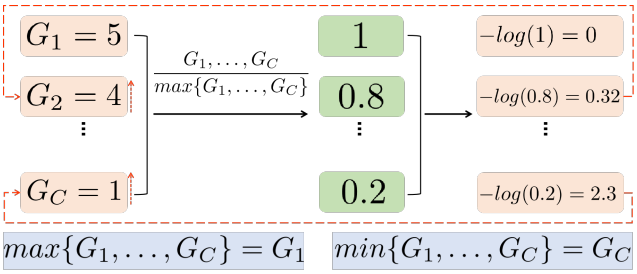

Figure 14. All curvatures smaller than G_1 gradually increase driven by the loss function, and the smaller the curvature, the greater the resulting loss.

converge to 1 and thus achieve curvature balance. However, the above operation violates the third design principle of curvature regularization, which is that the sum of curvatures of all perceptual manifolds tends to decrease. As shown in Fig 14, all curvatures smaller than G_1 gradually increase driven by the loss function, and the smaller the curvature, the greater the resulting loss. To solve this problem, we update each curvature to the inverse of itself before performing the maximum normalization of the curvature. Eventually, the curvature penalty term of the perceptual manifold M^i is denoted as $-\log(\frac{G_i^{-1}}{\max\{G_1^{-1},\ldots,G_C^{-1}\}})$. Further, the overall curvature regularization term is denoted as

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

As shown in Fig 15, the perceptual manifold with the smallest curvature produces no loss, and the larger the curvature, the larger the loss. $L_{Curvature}$ causes the curvature of all the perceptual manifolds to converge to the value with the smallest curvature while achieving equilibrium.

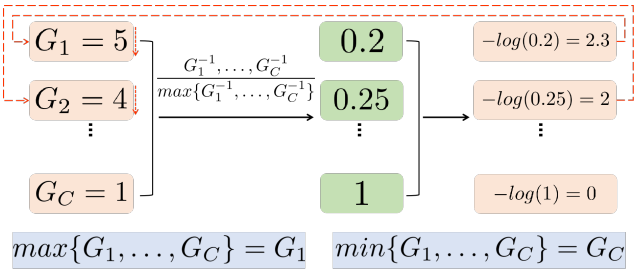

Figure 15. The perceptual manifold with the smallest curvature produces no loss, and the larger the curvature, the larger the loss.

Appendix G: Datasets and Implementation Details (Section 6)

Datasets and Evaluation Metrics

We conducted experiments on artificially created CIFAR-10-LT, CIFAR-100-LT [14], ImageNet-LT [14, 47], and real-world long-tailed iNaturalist2018 [55] to validate the effectiveness and generalizability of our method. 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.

- **CIFAR-10-LT** and **CIFAR-100-LT** are long-tailed datasets including five imbalance factors (IF = $10, 20, 50, 100, 200$) generated based on CIFAR-10 and CIFAR-100, respectively. The imbalance factor (IF) is defined as the value of the number of the most frequent class training samples divided by the number of the least frequent class training samples.

- **ImageNet-LT** is a long-tailed subset of ILSVRC 2012 with an imbalance factor of 256, which contains 1000 classes totaling $115.8k$ images, with a maximum of 1280 images and a minimum of 5 images per class. The balanced 50k images were used for testing.

- The **iNaturalist** species classification dataset is a large-scale real-world dataset that suffers from an extremely unbalanced label distribution. The 2018 version we selected consists of $437,513$ images from $8,142$ classes. The maximum class is $1,000$ images and the minimum class is 2 images (IF = 500).

- We use the **ILSVRC2012** split contains $1,281,167$ training and $50,000$ validation images. Each class of **CIFAR-100** contains 500 images for training and 100 images for testing.

Implementation Details

CIFAR-10/100-LT. To set up a fair comparison, we used the same random seed to make CIFAR-10/100-LT, and followed the implementation of [8]. We trained ResNet-32 by SGD optimizer with a momentum of 0.9, and a weight decay of 2×10^{-4}.

ImageNet-LT and iNaturalist2018. We use ResNext-50 [63] on ImageNet-LT and ResNet-50 [22] on iNaturalist2018 as the network backbone for all methods. And we conduct model training with the SGD optimizer based on batch size 256 (for ImageNet-LT) / 512 (for iNaturalist), momentum 0.9, weight decay factor 0.0005, and learning rate 0.1 (linear LR decay).

ImageNet and CIFAR-100. On ImageNet, we use random clipping, mixup [70], and cutmix [68] to augment the training data, and all models are optimized by Adam with batch size of 512, learning rate of 0.05, momentum of 0.9, and weight decay factor of 0.0005. On CIFAR-100, we set the batch size to 64 and augment the training data using random clipping, mixup, and cutmix. An Adam optimizer with learning rate of 0.1 (linear decay), momentum of 0.9, and weight decay factor of 0.005 is used to train all networks.

Appendix H: More Analysis of Dynamic Curvature Regularization

Here, we explored 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?

Recall that in Section 6.4, we trained multiple backbone networks on ImageNet and CIFAR-100. The features of all samples were extracted using ResNet-18 that was trained on ImageNet and CIFAR-100 with CE and with CE + CR, respectively, and the curvature of each perceptual manifold was calculated. The degree of imbalance is measured by the variance of the curvature of all perceived manifolds; the larger the variance, the more imbalanced the curvature. The experimental results are shown in Table 4, where the curvature of the perceptual manifolds represented by the ResNet-18 trained with curvature regularization is more balanced.

Table 4. The variance of the curvature of all perceptual manifolds.

	ImageNet	CIFAR-100
	ResNet-18	
CE	25.7	20.4
CE + CR	14.2 **(-11.5)**	11.8 **(-8.6)**
	VGG-16	
CE	27.4	23.5
CE + CR	13.8 **(-13.6)**	13.3 **(-10.2)**

Table 5. The Pearson correlation coefficient between the curvature of the perceptual manifold and the corresponding class accuracy.

	ImageNet	CIFAR-100
	ResNet-18	
CE	-0.583	-0.648
CE + CR	-0.257 **(+0.326)**	-0.285 **(+0.363)**
	VGG-16	
CE	-0.569	-0.635
CE + CR	-0.226 **(+0.343)**	-0.251 **(+0.384)**

We still use CE and CE + CR to train ResNet-18 on ImageNet and CIFAR-100, respectively, and then test the accuracy of two ResNet-18 on each class. The features of all samples were extracted using two ResNet-18 and the mean Gaussian curvature of each perceptual manifold was calculated. We calculated the Pearson correlation coefficients between the class accuracy and the curvature of the corresponding perceptual manifold for ResNet-18 trained with CE and with CE + CR, respectively. The experimental results are presented in Table 5, where it can be seen that the negative correlation between the mean Gaussian curvature of the perceptual manifold and the class accuracy decreases significantly after using curvature regularization. The same experiments are performed for VGG-16 and ResNet-18 in Tables 4 and 5.

Appendix I: Future Work

The model-independent measure of data difficulty

The performance of the model on different classes will vary. The bias is not introduced by the model structure, but by the characteristics of the data itself which affect the model's performance. Therefore, it is very important to propose model-independent measurements to characterize the data itself, and this work will greatly contribute to our understanding of deep neural networks. In this paper, the effect of volume, separation and curvature of data manifolds on the model bias is explored from a geometric perspective. It provides a new direction for future work, namely the geometric analysis of deep neural networks.

The geometric perspective of data classification

Natural datasets have intrinsic patterns that can be generalized to the manifold distribution principle: the distribution of a class of data is close to a low-dimensional manifold. Data classification can be regarded as the unwinding and separation of manifolds. When a data manifold is entangled with other perceptual manifolds, the difficulty of classifying that manifold increases. Typically, a deep neural network consists of a feature extractor and a classifier. Feature learning can be considered as manifold unwinding, and a well-learned feature extractor is often able to unwind multiple manifolds for the classifier to decode. In this view, all factors about the manifold complexity may affect the model's classification performance. Therefore, we suggest that future work can explore the inter-class long-tailed problem from a geometric perspective.

The geometric perspective of object detection

In the field of object detection, it is often encountered that although a class does not appear frequently, the model can always detect such instances efficiently. It is easy to observe that classes with simple patterns are usually easier to learn, even if the frequency of such classes is low. Therefore, classes with low frequency in object detection are not necessarily always harder to learn. We believe that it is a valuable research direction to analyze the richness of the instances contained in each class, and then pay more attention to the hard classes. The dimensionality of all images or feature embeddings in the image classification task is the same, which facilitates the application of the semantic scale proposed in this paper. However, the non-fixed dimensionality of each instance in the field of object detection brings new challenges, so we have to consider the effect of dimensionality on the semantic scale, which is a direction worthy of further study.