Hierarchical Convolutional Neural Network with Knowledge Complementation for Long-Tailed Classification

HONG ZHAO*, ZHENGYU LI, WENWEI HE, and YAN ZHAO, School of Computer Science, Minnan Normal University; Key Laboratory of Data Science and Intelligence Application, Minnan Normal University, Zhangzhou, China

Existing methods based on transfer learning leverage auxiliary information to help tail generalization and improve the performance of the tail classes. However, they cannot fully exploit the relationships between auxiliary information and tail classes and bring irrelevant knowledge to the tail classes. To solve this problem, we propose a hierarchical CNN with knowledge complementation, which regards hierarchical relationships as auxiliary information and transfers relevant knowledge to tail classes. First, we integrate semantics and clustering relationships as hierarchical knowledge into the CNN to guide feature learning. Then, we design a complementary strategy to jointly exploit the two types of knowledge, where semantic knowledge acts as a prior dependence and clustering knowledge reduces the negative information caused by excessive semantic dependence (i.e., semantic gaps). In this way, the CNN facilitates the utilization of the two complementary hierarchical relationships and transfers useful knowledge to tail data to improve long-tailed classification accuracy. Experimental results on public benchmarks show that the proposed model outperforms existing methods. In particular, our model improves accuracy by 3.46% compared with the second-best method on the long-tailed tieredImageNet dataset.

Additional Key Words and Phrases: long-tailed classification, deep learning, knowledge transfer, hierarchical relationship

1 INTRODUCTION

The visual world is natively long-tailed and imbalanced [25]. In deep learning, the class frequencies follow a long-tailed distribution [16]: A few classes (i.e., head classes) occupy most of the samples, while a large number of classes (i.e., tail classes) have scarce samples. Long-tailed classification has a range of practical applications, such as face recognition [34], medical diagnosis [14], and fault analysis [9]. The goal is not only to handle head classes with massive amounts of data but also to classify tail classes with limited data.

Recently, significant advances in deep convolutional neural networks (CNNs) have been made in computer vision fields such as image classification [17] and object detection [27]. Increasing attention is being focused on utilizing CNNs to learn high-quality class features for long-tailed classification. In contrast to vision tasks trained on balanced data, long-tailed data poses several great challenges to CNNs. (1) Few-shot learning difficulty for tail classes [51]: Tail classes with limited training data are not compatible with the data-hungry limitations of CNNs. This seriously decreases the representative ability of deep learned features. (2) Class imbalance difficulty for overall classes [32]: The feature space of head classes is larger than that of tail classes, while the decision boundary is biased towards head classes. When head features dominate tail features, the tail classes are usually overwhelmed by

*Corresponding author.

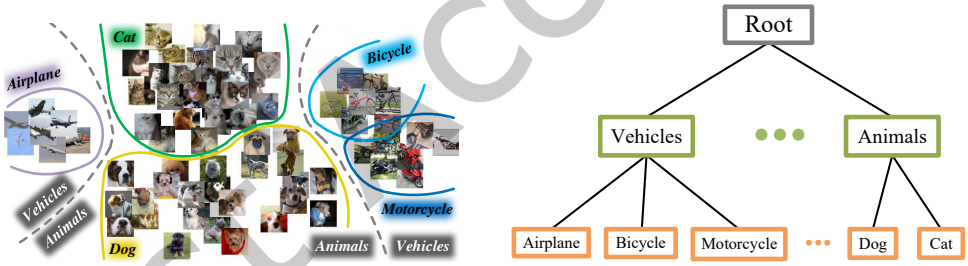
Authors' address: Hong Zhao, hongzhaocn@163.com; Zhengyu Li; Wenwei He; Yan Zhao, School of Computer Science, Minnan Normal University; and Key Laboratory of Data Science and Intelligence Application, Minnan Normal University, Zhangzhou, China.

ACM 1556-4681/2024/3-ART
https://doi.org/10.1145/3653717

the head classes and misclassified as the head classes. These problems prevent CNNs from learning discriminative features, especially within tail classes, which leads to unsatisfactory performance in long-tailed classification.

To alleviate the challenges of long-tailed data, existing work typically adopts data re-sampling [16], loss re-weighting [34, 38], and transfer learning from head classes to tail classes [17, 48]. Transfer learning achieves satisfactory performance mainly by transferring head knowledge to tail features. However, there are two main limitations to transfer learning: (1) The visual similarity between head and tail classes are not fully explored. Thus, new tail features based on head knowledge have no meaning for interpretation, *e.g.*, the head and tail classes have no similar visual appearance. (2) When the head features are different from the tail features, the effect of knowledge transfer is quite limited. Thus, the transfer learning method introduces irrelevant knowledge to the tail classes and reduces the knowledge representative ability of tail features.

Hierarchical super-classes contain strong similarities among classes and have been widely used in classification tasks [50]. These similarities are regarded as auxiliary information to provide extra guidance for long-tailed scenarios and commonly appear in two forms: semantic [5] and clustering [22] super-classes. Semantic knowledge involves relationships, hierarchies, and similarities between things [46]. A semantic super-class groups semantically similar classes into a super-class set, which helps to learn intra-class compact and inter-class separable features. Figure 1 visualizes the clustering and semantic relationships of several classes to explain the relationship between class features and semantic similarity. (1) Semantic similarity: *Bicycle* features are closer to *Motorcycle* because both classes are attached to the same semantic super-class *Vehicles*. (2) Semantic dissimilarity: *Cat* features are different from *Bicycle* features as these classes belong to different semantic super-classes We use the super-class feature to co-represent similar classes and transfer correlative super-class knowledge to tail classes. The semantic super-class knowledge has physical meaning for interpretation and mitigates the poor generalization of CNNs to tail classes.



(a) Visualization of the clustering features. (b) Visualization of the class semantic relationships.

Fig. 1. Clustering and semantic relationship visualizations of the five classes *Cat*, *Dog*, *Bicycle*, *Motorcycle*, and *Airplane*. From a data perspective, Figure 1(a) shows the clustering results of these five classes. From a label perspective, Figure 1(b) illustrates the semantic hierarchical relationship of these five classes. *Vehicles* is a hypernym that includes *Airplane*, *Motorcycle* and *Bicycle*. *Bicycle* and *Motorcycle* are similar in shape, so they can be considered as the same super-class. *Bicycle* and *Airplane* are different in shape, but both serve as transportation and can be considered to be in the same super-class.

However, semantically related classes are not always similar in terms of visual features, and relying only on hierarchical semantic relationships is insufficient for a model to distinguish between difficult samples since semantic gap problems group dissimilar classes into the same super-class. Excessive dependence on semantic similarities introduces negative features, and it is difficult for a super-class feature to co-represent visually dissimilar classes. As shown in Figure 1, *Bicycle* is dissimilar to *Airplane* despite belonging to the same semantic super-class *Vehicles*.

Thus, jointly learning *Bicycle* and *Airplane* features hurt the knowledge representations of super-class *Vehicles*. In contrast to semantic super-classes, clustering super-classes mine underlying similarities from the data itself [26] and fully explore the relationships between auxiliary super-classes and samples. This reflects distinguishing features among semantically similar classes that are not obtained by semantic super-classes. We further use clustering super-classes to compensate for the blind semantic dependency and improve the discrimination of knowledge representations, thus bridging the semantic gap problem.

In this paper, we propose a hierarchical CNN via knowledge complementation (HCKC) to transfer correlative knowledge to tail classes. First, we establish two hierarchical super-classes to correlate tail classes and train a hierarchical CNN to extract semantic and clustering super-class knowledge representations. This gives the class relationships a hierarchical interpretation and provides super-class representations as auxiliary information for tail feature learning. Then, we transfer the semantic knowledge to tail data to learn correlative features and use the auxiliary knowledge to facilitate the representative ability of tail classes. Meanwhile, we provide clustering knowledge to compensate for the adverse semantic representations caused by semantic gaps. The complementary strategy involving two super-classes fully correlates auxiliary information with tail classes and learns useful knowledge to improve CNN training.

We evaluate HCKC on six long-tailed datasets, including two real-world datasets. We conduct an ablation study to verify each design in our model. In addition, we demonstrate how to overcome semantic gap problems with our knowledge complementary strategy. The HCKC model is effective compared with existing methods, including several transfer learning methods, indicating that correlative knowledge facilitates the transfer learning effect. Compared with the second-best method, HCKC had 3.46% better accuracy on the long-tailed dataset tieredImageNet. The datasets and the our code for the proposed algorithms are available at http://github.com/fhqxa/lzy_HCKC.

The main contributions of this paper are as follows:

(1) We explore both clustering and semantic relationships, providing a physical explanation for migrating clustering and semantic knowledge transfer.

(2) We prove that clustering relationships can help solve the semantic gap problem in hierarchical relationships and jointly guide long-tail classification from both data and label aspects.

(3) We propose a complementary strategy to mediate semantic and clustering relationships, thereby helping tail classes learn more diverse knowledge and expand their feature diversity.

The remainder of this paper is organized as follows. In Section 2, we introduce related work. In Section 3, we present our model. In Section 4, we describe the experimental datasets and implementation details. In Section 5, we conduct an ablation study and compare its performance with state-of-the-art methods. Section 6 concludes the paper.

2 RELATED WORK

In this section, we briefly review related work on long-tailed classification and hierarchical super-class learning.

2.1 Long-tailed classification

Data with a long-tailed distribution is a long-standing issue in machine learning. The key challenge is to classify tail data and correct any imbalanced problem. Given that deep learning has reached satisfying performance with balanced data, increasing research is being focused on tailoring CNNs to learn discriminative features for long-tailed classification tasks. Most existing methods based on CNNs are divided into re-sampling, re-weighting, and transfer learning methods.

Re-sampling. Data re-sampling is a natural approach to artificially remedying a lack of information. It can augment tail samples and alleviate feature imbalances. Two representative re-sampling methods are over-sampling

and under-sampling. (1) Over-sampling repeats tail samples [1], which easily causes model over-fitting. Consequently, Chawla et al. [4] raised a synthetic minority over-sampling technique (SMOTE) to generate samples in the feature space. Several variants of SMOTE based on clustering have since been developed [44]. (2) Under-sampling discards abundant head data [33], which certainly impairs the generalization ability of deep networks. Thus, Lin et al. [21] proposed a clustering-based under-sampling technique to reduce the risk of removing head data. Further, some re-sampling methods incorporating transfer learning have been presented. Kim et al. [17] suggested a transfer learning-based re-sampling method and translated the diversity of the head information to augment tail data. Moreover, Zhou et al. [48] explored a bilateral-branch network and fused the features of a re-sampling branch into imbalanced data.

Re-weighting. Also known as cost-sensitive learning, re-weighting modifies loss functions in the CNN training process. This usually assigns larger tail weights than head classes, aiming to be inversely proportional to class frequencies [13]. Subsequently, Lin et al. [20] reduced the weights of head samples and designed a focal loss approach to dealing with extremely imbalanced problems. Similarly, Cao et al. [2] raised the margin of tail classes and proposed a label-distribution-aware margin loss technique. To decrease data overlap, Cui et al. [6] further allocated weights as the number of samples increases according to the effective inverse number of classes. When it comes to more practical scenarios, tail classes are easily overwhelmed by head classes. Thus, Tan et al. [35] proposed using equalization loss to ignore negative gradients for tail classes and protect them from being at a disadvantage during network learning. Further, Tan et al. [34] introduced a gradient-guided reweighting mechanism to improve equalization loss performance. Subsequently, Wang et al. [38] design the seesaw loss technique for dynamically re-balancing the gradients of positive and negative samples for each class.

Transfer learning. Transfer learning enriches tail knowledge by using auxiliary information and related tasks [11]. Wang et al. [40] leveraged a meta-network to encode head knowledge and progressively transferred the knowledge to tail classes. Further, Liu et al. [23] introduced a dynamic meta-embedding technique to transfer memory features to tail classes. Kim et al. [17] explored a data augmentation approach by transferring the diversity of head information to tail classes. Unlike the above transfer learning methods, which directly transfer head knowledge, our method utilizes semantics and clustering relationships to learn the head class knowledge that is relevant to tail classes and extract representations that are helpful for tail classification.

2.2 Hierarchical super-class learning

A hierarchical relationship appears in a hierarchical tree and usually consists of two levels of abstraction: the super-class and target class. Super-classes reflect knowledge concepts and correlative information among the target classes used in semantic and clustering super-classes.

Semantic super-class. Humans present a powerful generalized capability in recognizing examples, which follows an abstract-to-concrete mechanism [37]. Inspired by human intelligence, the semantic hierarchy is constructed to exploit abstract information from semantic super-classes. One of the most popular semantic hierarchical trees is WordNet [28]. This groups words into synonyms and highlights the different semantic relationships among them. Deng et al. [8] maximized information gain in a classification system and maintained a small error rate by using semantic super-classes. Similarly, Ge et al. [10] integrated semantic super-classes into cross-entropy loss-based deep networks and considered the risk of misclassification. To further improve classification performance, Inoue et al. [15] learned semantic super-class details using a hierarchical CNN topology to improve target accuracy. Chen et al. [5] made full use of semantic super-class information and proposed a hierarchy-aware label semantics matching network. However, fully trusting the label semantic superclass information leads to the semantic gap problem. To avoid this problem, our method introduces clustering relationships to assist labeling of semantic information from the data-feature level.

Clustering super-class. A typical example is the hierarchical tree method, which learns a tree structure by recursively clustering target classes into disjoint groups (*i.e.*, super-classes) [49]. The hierarchical structure places the targets that are easily confused by classifiers into groups with the same abstract concepts, making the representations associated with the clustering super-classes easily learnable. Chang et al. [3] combined representation learning with clustering and developed a deep self-evolution clustering algorithm. Xu et al. [43] combined contrastive learning with a deep clustering framework that can learn to distinguish representations. In contrast to single deep learning models, Zhong et al. [47] divided a task into multiple cluster subtrees and designed a recurrent network to learn local structural relationships. Liu et al. [22] provided an adaptively neural clustering architecture by grouping data points in a cluster-wise view rather than a point-wise view, which sufficiently mines the hidden structure among data. With the advent of large-scale data, Naumov et al. [26] proposed a practical clustering algorithm to address the challenge of scaling up hierarchical clustering to large datasets. Unlike methods that directly cluster the dataset, our method is based on the characteristics of long-tail data. For tail classes with insufficient samples, we exploit clustering methods to cluster tail super-classes, which assists the tail in obtaining more knowledge.

3 KNOWLEDGE COMPLEMENTARY STRATEGY VIA HIERARCHICAL CNN

In this section, we describe a hierarchical CNN with knowledge complementary strategy (HCKC) for long-tailed classification tasks. We present the general framework of the proposed HCKC and introduce its components in detail.

3.1 Framework of the HCKC model

The general framework of the HCKC model is shown in Figure 2, and consists of three main components.

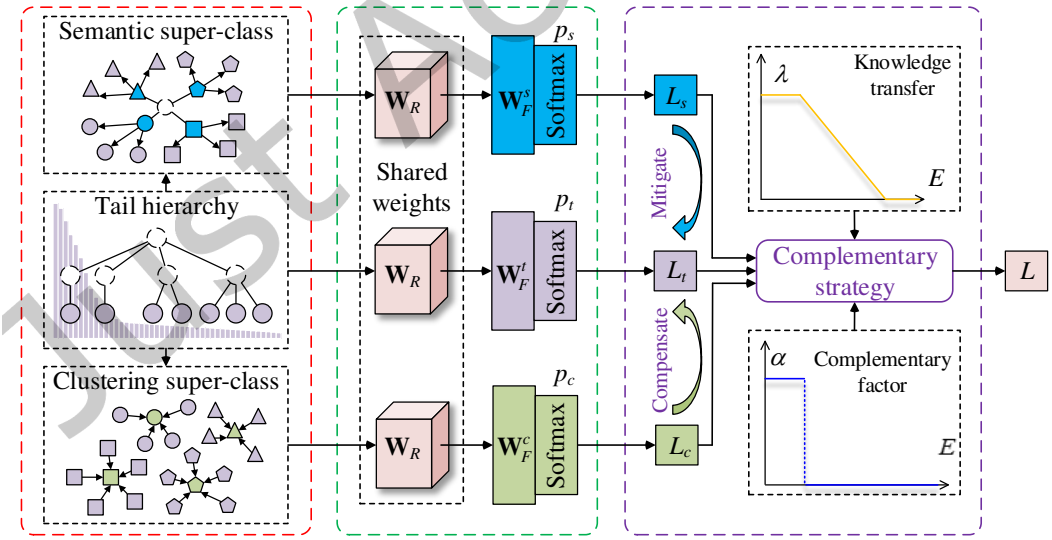


Fig. 2. Framework of the HCKC model.

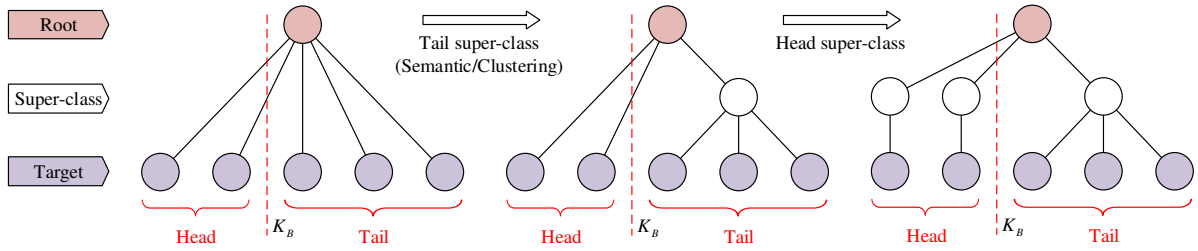


Fig. 3. An example of the tail hierarchical structure.

(1) Tail hierarchical super-class establishment: We build two hierarchical relationships (semantic and clustering hierarchies) and obtain two super-class labels (semantic and clustering super-classes). The two super-class relationships are regarded as auxiliary information to assist the feature learning of target classes. Meanwhile, we tailor the two relationships to pay attention to the tail target classes and boost long-tailed classification accuracy.

(2) Hierarchical CNN construction: We construct a hierarchical CNN divided into shared-representation and task-specific layers. Input data with target, semantic, and clustering class labels are fed into the hierarchical CNN to extract different knowledge representations by corresponding learning tasks.

(3) Knowledge complementary strategy: We design a knowledge complementary strategy to train the hierarchical CNN. The training process transfers semantic representations and then clustering representations to the tail target classes.

3.2 Tail hierarchical super-class establishment

We consider a classification task with N target classes from a long-tailed dataset $\mathcal{D}$ $\{x_i, y_i^t\}_{i1}^{K}$, where x_i denotes an input sample and y_i^t denotes a target label. Let K $_{n1}^{N}$ K_n denote the sample size, where K_n is the number of n^{th} class, $n \in \{1, 2, ..., N\}$. We assume that classes are sorted by cardinality in decreasing order, *i.e.*, $K_1 \geq K_2 \geq ... \geq K_N$. Let K_B denote a boundary for separating head classes (*i.e.,* $K_1 \geq K_2 \geq ... \geq K_{B-1}$) from tail classes (*i.e.,* $K_B \geq ... \geq K_{N-1} \geq K_N$), where $N - B$ denotes the number of tail classes. Long-tailed classifications aim to improve performance in tail classes.

We use super-class label information to help tail target learning with hierarchical relationships. The conventional hierarchical tree applies its whole hierarchical structure to all classes for a balanced dataset. In contrast to a balanced distribution, we attempt to utilize the hierarchical structure differently. The proposed structure takes account of tail classes instead of directly using the original hierarchical tree for all classes. The structure makes the CNN pay greater attention to tail classes than head classes. An example of the tail hierarchical structure is illustrated in Figure 3. For head classes, we generate the super-classes and keep their class information consistent with the target classes. We employ the hierarchical relationship of the tail classes to obtain their corresponding super-classes. In this way, we obtain two types of super-class: tail semantic and tail clustering super-classes.

Tail semantic super-classes. We employ the semantics of data labels to integrate prior knowledge about target class relationships into visual learning. Classes are semantically similar under the same semantic super-class. Our motivation is that semantically similar classes are often similar in their appearance, such as color, texture, or shape. Therefore, the semantic labels group similar features together. For example, *Bicycle* and *Motorcycle* belong to *Vehicles* in terms of semantic relation; thus, they have a similar visual appearance. We obtain a tail semantic super-class set according to WordNet. Then, we obtain all semantic super-classes y^s using the proposed tail hierarchical structure. However, a semantic gap inherently exists in various application scenarios and classification tasks. This problem has been studied extensively in content-based image retrieval [24, 30]. Humans understand

images based on high-level semantic concepts, while machines compute images based on low-level features. This gap in understanding between the semantic labels and image features hampers user requirements.

We define the semantic gap as a visually dissimilar problem under semantic consistency. Semantically similar classes may present visually distinguishable appearances, called semantic gaps. Semantic gaps lead to the grouping of visually different features into adjacent domains in the feature space. Although *Bicycle* and *Airplane* belong to *Vehicles* in terms of semantic relations, both of them have distinct visual appearances, which degrades the representation learning of deep networks. Therefore, we use a clustering technique to measure the class similarity differently.

Tail clustering super-class. Given a similarity graph $\mathbf{G} \ \mathbf{V}, \mathbf{E}$ that contains the similarity information for each two tail target classes, where each vertex $\mathbf{v}_n \in \mathbf{V}$ denotes a tail target class and each edge $e_{nn'} \in \mathbf{E}$ denotes the degree of similarity between two connected vertices $\mathbf{v}_n$ and $\mathbf{v}_{n'}$. For the n^{th} tail target class, the corresponding vertex $\mathbf{v}_n$ is vectorized and represented by the centroid of all its features, which is formulated as

$$\mathbf{v}_n \ \frac{1}{K_n} \sum_{i1}^{K_n} f x_i, \tag{1}$$

where an input sample x_i is fed into a representation learning layer of network $f\cdot$ to extract a feature representation and $n \in \{B, ..., N-1, N\}$. The edge $e_{nn'}$ is computed by Gaussian similarity:

$$e_{nn'} \ e^{-\frac{\|\mathbf{v}_n - \mathbf{v}_{n'}\|^2}{2\sigma^2}}, \tag{2}$$

where σ is a scaling factor that normalizes the value of $e_{nn'}$ to $0, 1$. We apply NCut technique [39] to cut the similarity graph into several sub-graphs, which are represented to optimize

$$\begin{aligned} &\min_{C_1, C_2, ..., C_L} Tr\mathbf{H}'\mathbf{P}\mathbf{H}, \\ &s.t. \ \mathbf{H}'\mathbf{D}\mathbf{H} \ \mathbf{I}, \end{aligned} \tag{3}$$

where $\mathbf{P} \ \mathbf{D} - \mathbf{G}$ denotes Laplacian matrix, $\mathbf{D}$ is the degree matrix of $\mathbf{G}$, and $\mathbf{I}$ is the identity matrix. For the matrix $\mathbf{H}$, its element h_{nl} is written as

$$h_{nl} \ \begin{cases} \frac{1}{|C_l|} & \text{if } \mathbf{v}_n \in C_l, \\ 0 & \text{otherwise}, \end{cases} \tag{4}$$

where C_l denotes the l^{th} super-class label, $l \in \{1, 2, ..., L\}$, and L is the number of clustering super-classes. This optimization objective is approximated by the eigenvector of $\mathbf{P}$ associated with the second-smallest eigenvalue. In this way, we generate tail clustering super-class sets. Then, we obtain all clustering super-classes y^c from the proposed tail hierarchical structure.

3.3 Hierarchical CNN construction

We design a hierarchical CNN architecture to train the data with super-class and target labels. The hierarchical CNN is used to perform target, semantic, and clustering learning tasks. Then, the input data is fed into shared representation learning layers and task-specific learning layers in the hierarchical CNN.

The input data for the representation learning comes from the target task. The target task retains its original features and long-tailed data distribution. The tail hierarchical structure attempts to alleviate the imbalanced problem by having a relatively uniform distribution, and its input data comes from two super-class tasks: semantic and clustering tasks. For super-class tasks, each sample converges to the same super-class. Thus, the super-classes ease a data-hungry limitation of representation learning and extract generalized knowledge. We leverage the shared

representation learning layers to train the data from different tasks. The representation learning layers contain convolutional layers and pooling layers for a CNN. The shared representation learning is represented as

$$\mathbf{Z}_R \quad fx, \mathbf{W}_R, \tag{5}$$

where $\mathbf{Z}_R$ denotes the output of the representation learning layers, and $fx, \mathbf{W}_R$ is implemented by the representation learning layers with parameter $\mathbf{W}_R$. The shared weights use related information in different tasks, which helps the CNN learn a more generalized feature.

Task-specific learning layers. A fully connected (FC) layer of the target task is used to weight the output $\mathbf{Z}_R$ of the representation learning. Output logits of the target FC layer are formulated as

$$\mathbf{Z}_F \quad \mathbf{W}_F\mathbf{Z}_R \quad \mathbf{b}, \tag{6}$$

where $\mathbf{Z}_F$ is a predicted output, *i.e.*, $z_1, z_2, ..., z_N{}^{\mathrm{T}}$, $\mathbf{W}_F$ is the weight vector of the target FC layer, and $\mathbf{b}$ is a bias. For each target class $n \in \{1, 2, ..., N\}$, the n^{th} class probability is computed by softmax function:

$$p_n \quad \frac{e^{z_n}}{\sum_{n1}^{N} e^{z_n}}. \tag{7}$$

The output probability distribution of the target task is denoted as $\mathbf{p} \quad p_1, p_2, ..., p_N{}^{\mathrm{T}}$, where $p_n \in 0, 1$. In our hierarchical CNN, there are three specific tasks to be learned: target, semantic, and clustering tasks. We can obtain label sets of the target, semantic, and clustering tasks, denoted as y^t, y^s, and y^c, respectively. Each task is attached to the specific FC layer to extract discriminative knowledge, which is denoted as $\mathbf{W}_F^t$, $\mathbf{W}_F^s$, and $\mathbf{W}_F^c$ for target, semantic, and clustering tasks, respectively. Thus, the estimated class probabilities are written as $\mathbf{p}_t$, $\mathbf{p}_s$, and $\mathbf{p}_c$ for target, semantic, and clustering classes, respectively. The task-specific layers learn different discriminative features and facilitate the construction of the hierarchical CNN.

Different features are extracted by shared representation and task-specific layers. We regard two super-class features as auxiliary information to guide knowledge transfer. Then, the two types of auxiliary knowledge are adaptively adjusted by a knowledge complementary strategy on subsequent loss layers.

3.4 Knowledge complementary strategy

We transfer class correlative knowledge to target features for alleviating the long-tail problem. Loss function $\mathcal{L}$ of HCKC is formulated as

$$\mathcal{L} \quad \lambda\mathcal{L}_k \quad 1 - \lambda\mathcal{L}_t, \tag{8}$$

where λ is a weighted parameter by a transfer curve and $\mathcal{L}_t$ denotes a target loss. Here $\mathcal{L}_k$ works as a tunable trade-off factor between semantic and clustering knowledge representations. The loss determines $\mathcal{L}_k$ by two weighted terms, as

$$\mathcal{L}_k \quad \alpha\mathcal{L}_s \quad \beta\mathcal{L}_c, \tag{9}$$

where $\mathcal{L}_s$ denotes a semantic loss, and $\mathcal{L}_c$ denotes a clustering loss. The weighted semantic term α mitigates the poor generalization of tail targets. The weighted clustering term β compensates for the adverse effect of semantic knowledge. Generally, the two weighted terms jointly adjust the complementary strategy, and loss $\mathcal{L}$ is our optimization goal.

Weighted semantic term. Semantic super-classes correlate tail classes with semantic similarity and learn a generalized representation. We leverage the semantic representation to alleviate the poor training. The number of training epochs adaptively adjusts the weighted semantic term:

$$\alpha \quad \begin{cases} 1 & \text{if } 0 < E \leq E_c, \\ 0 & \text{otherwise}, \end{cases} \tag{10}$$

where E is the current epoch, and E_c denotes the number of cluster training epochs. The weighted semantic term determines that the semantic knowledge is first applied. There are two reasons for this: (1) By training the CNN from a random initialization point, extracted features are usually represented poorly and are difficult to generalize to test samples. (2) The challenge of random initialization is aggravated by data with a long-tailed distribution. Instead, semantic knowledge independent of abundant data is used as prior knowledge, which is not degraded by rare data with a long-tailed distribution.

Weighted clustering term. The weighted semantic term makes the CNN train on semantically similar tail classes. Nevertheless, it may lead to a visual dissimilarity under the same super-class. Consequently, the poor representation degrades the training on tail classes. We propose a weighted clustering term that clusters tail data according to the visual similarity and sample distances in the feature space. The weighted clustering term β is calculated as $1 - \alpha$, which determines that clustering knowledge is applied rather than semantic knowledge.

The tail samples are clustered into the same visually similar super-classes when $E > E_c$. The weighted clustering term reduces tail class dependence on semantic knowledge and prevents the semantic gap from degrading the tail features. It aids the network in obtaining valuable and robust representations. Otherwise, the semantic knowledge is applied when β 0.

Knowledge transfer. A knowledge transfer strategy is proposed to shift the learning focus between the super-class and target tasks by controlling the weight λ. It is designed to learn correlative information first and then gradually pay attention to tail target data. In the training process, the loss of the super-class task is multiplied by λ, and the target loss is multiplied by $1 - \lambda$. Here, λ is automatically generated according to the training epoch, which is calculated by

$$\lambda \begin{cases} 1 & 0 < E \leq E_c, \\ 1 - \dfrac{E}{E_{max}} & E_c < E \leq E_c \ E_s, \\ 0 & \text{otherwise}, \end{cases} \tag{11}$$

where E_{max} denotes the number of total training epochs and E is the current epoch. In this respect λ descends linearly with knowledge transfer, where the role of the super-class and target tasks are symmetric. Parameter λ determines that the CNN attention is turned from the super-classes to the target classes.

In the experiment, $\mathcal{L}$ is a simple usage for the cross-entropy loss that we consider as a baseline for HCKC. The HCKC loss in Equation (8) becomes

$$\begin{aligned} \mathcal{L} \ & \lambda \mathcal{L}_k \ 1 - \lambda \mathcal{L}_t \\ & \lambda\alpha \mathcal{L}_s \ \beta \mathcal{L}_c \ 1 - \lambda \mathcal{L}_t \\ & \lambda\alpha \mathcal{L}_{CE} x, \mathbf{W}_R, \mathbf{W}_F^s \ \beta \mathcal{L}_{CE} x, \mathbf{W}_R, \mathbf{W}_F^c \ 1 - \lambda \mathcal{L}_{CE} x, \mathbf{W}_R, \mathbf{W}_F^t, \end{aligned} \tag{12}$$

where $\mathcal{L}_{CE}\cdot$ is the standard cross-entropy loss function.

The training process for the hierarchical CNN is expressed in Algorithm 1. Line 1 suggests that a CNN is trained from a random initialization. Lines 3 to 11 describe the loss calculated via the knowledge complementary strategy. Line 12 means that the learning rate is altered. Line 13 implies that the CNN parameters are updated.

The test process is independent of CNN training and utilizes plain learning without any algorithms to evaluate the performance of the trained CNN on test datasets. The experimental datasets and settings are described in detail in Section 4.

4 EXPERIMENTAL SETTINGS

This section introduces datasets and implementation details used in our experiments. Then, we introduce several state-of-the-art methods and evaluation measures.

Algorithm 1 Long-tailed classification learning via HCKC

Input: A dataset $\mathcal{D}$ $\{x_i, y_i^t\}_{i1}^K$ with N target classes, where x denotes input data, y^t denotes the target labels, and K denotes the sample size of the dataset. Parameter y^s denotes the semantic labels, and y^c denotes the clustering labels. The number of total training epochs is E_{max}, E_s is the number of semantic training epochs, and E_c is the number of cluster training epochs.

Output: A parameterized CNN $\mathbf{W}$ $\mathbf{W}_R; \mathbf{W}_F^s; \mathbf{W}_F^c; \mathbf{W}_F^t$.

Initialize CNN parameters $\mathbf{W}$ $\mathbf{W}_R; \mathbf{W}_F^s; \mathbf{W}_F^c; \mathbf{W}_F^t$ randomly;
for E $1 : E_{max}$ **do**
  if $E \leq E_c$ **then**
    $\mathcal{L}\mathbf{W} \leftarrow L_{CE}x, \mathbf{W}_R, \mathbf{W}_F^s$ with λ 1 and α 1;
  else if $E \leq E_c$ E_s **then**
    Update λ by Equation (11);
    Update clustering super-class set by Equation (3);
    $\mathcal{L}\mathbf{W} \leftarrow \lambda L_{CE}x, \mathbf{W}_R, \mathbf{W}_F^c$ $1 - \lambda\mathcal{L}_{CE}x, \mathbf{W}_R, \mathbf{W}_F^t$ with α 0;
  else
    $\mathcal{L}\mathbf{W} \leftarrow L_{CE}x, \mathbf{W}_R, \mathbf{W}_F^t$ with λ 0;
  end if
  Update learning rate r;
  Update $\mathbf{W} \leftarrow$ A stochastic gradient descent optimizer $\mathbf{W} - r\nabla_{\mathbf{W}}\mathcal{L}\mathbf{W}$;
end for

4.1 Datasets

We evaluate our model on six long-tailed datasets: long-tailed CIFAR-10 (LT-CIFAR-10), long-tailed CIFAR-100 (LT-CIFAR-100)[1], long-tailed TinyImageNet (LT-TYIN)[2], long-tailed tieredImageNet (LT-TDIN)[3], SUN-324[4], and iNaturalist[5]. Table 1 provides descriptions of certain datasets. This structure is constructed by the semantic dependency between classes in WordNet [15, 28].

LT-CIFAR-10 and LT-CIFAR-100. Both LT-CIFAR-10 and LT-CIFAR-100 [19] consists of 10 and 100 classes with a uniform distribution. We utilize long-tailed versions as the same as those used in [2, 17]:

(1) Denote imbalance ratio as ρ K_1K_N.

(2) K_n between K_1 and K_N follows an exponential decay across class n. Imbalance ratios ρ that we use in experiments are 10, 50 and 100.

LT-TYIN. TinyImageNet is a subset of ImageNet2012 [31] and contains 200 classes. Each class has 500 training images and 50 test images.

LT-TDIN. tieredImageNet [29] is a subset of ImageNet2012 [31]. This groups 608 classes into 34 semantic super-classes by the semantic hierarchy [7].

SUN-324. The extensive Scene UNderstanding (SUN) database [41, 42] contains 397 classes. We remove multi-label classes on SUN and retain 324 classes. Figure 4(a) illustrates distributions of classes for SUN-324. The imbalance ratio of SUN-324 is 27.9. The 324 classes are grouped into 15 semantic super-classes.

[1] https://www.cs.toronto.edu/kriz/cifar.html
[2] https://aistudio.baidu.com/datasetdetail/201309
[3] http://cs231n.stanford.edu/tiny-imagenet-200.zip
[4] https://sun3d.cs.princeton.edu/
[5] https://github.com/visipedia/inat_comp/tree/master/2018

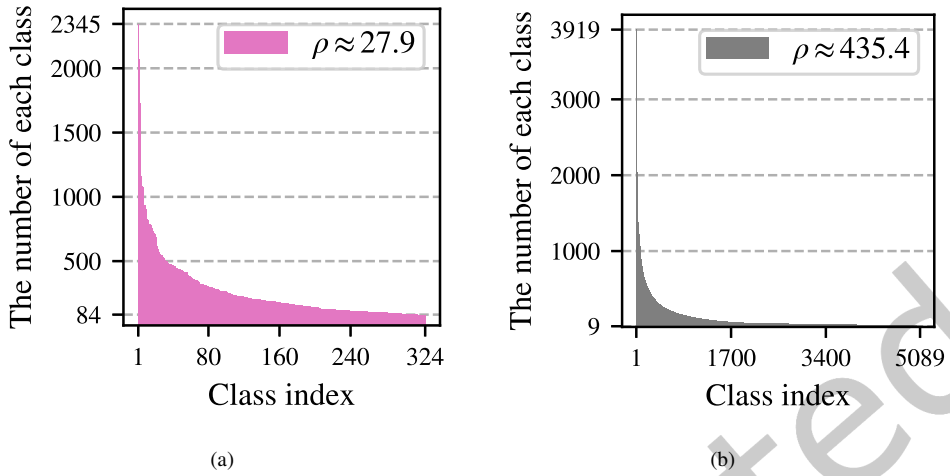


Fig. 4. Histograms on number of training samples per class on different datasets: (a) SUN-324 and (b) iNaturalist.

iNaturalist. The 2017 version of iNaturalist [36] contains 675,170 images with 5,089 classes. Figure 4(b) illustrates distributions of classes for iNaturalist. The imbalance ratio of iNaturalist is 435.4. We use the official hierarchy and split containing 579,184 training and 95,986 test images.

Table 1. Descriptions of the experimental datasets.

Dataset	Coarse-grained class	Fine-grained class	Dataset size	Imbalance way
LT-CIFAR-10	2	10	60,000	Artificial
LT-CIFAR-100	20	100	60,000	Artificial
LT-TYIN	-	200	100,000	Artificial
LT-TDIN	34	608	779,165	Artificial
SUN-324	15	324	85,147	Real-world
iNaturalist	13	5089	579,184	Real-world

For all of these experimental datasets: (1) A semantic relationship from WordNet organizes classes. (2) Samples are single-labeled for both classes and semantic super-class classes. (3) We regard 80% few classes as tail classes and the rest as head classes.

4.2 Implementation details

The code is developed using the PyTorch framework. We conduct experiments on a single NVIDIA GeForce RTX 2080 Ti GPU. We follow the basic settings proposed in [48] on each dataset.

Training details. We adopt a data augmentation technique or its horizontal flip with 4 pixels padded and ensure that input images have the size of 32×32 either by cropping or re-sizing. Residual network [12] with 32 layers (*i.e.*, ResNet-32) is adopted to train every model. We employ standard mini-batch stochastic gradient descent with a momentum of 0.9, weight decay of 2×10^{-4} and batch size of 128.

Details on HCKC. During training, we divide the training process into three stages: semantic, clustering, and target learning. Clustering knowledge descends linearly in the clustering learning stage, while learning attention is individually paid to other stages to focus on corresponding tasks. The learning rate is set to 0.1 across the semantic stage with 55 epochs and the clustering stage has 45 epochs. For the target stage with 200 epochs, the initial learning rate is set to 0.1, where we adopt linear warm-up learning rate schedule in the first five epochs. The learning rate is decayed by 0.01 in the last 40 epochs and again in the last 20 epochs.

4.3 Comparison methods

We compare the HCKC model with three methods: re-sampling, re-weighting, and transfer learning.

Re-sampling methods. (1) re-sampling (RS) [16]: Balancing the dataset for equal numbers in each class; (2) deferred re-sampling (DRS) [2]: RS waits until the later stage of training; (3) synthetic minority over-sampling (SMOTE) [4]: A variant of RS with data augmenting based on feature space.

Re-weighting methods. (1) cross-entropy loss (CE): A baseline training and each training sample have an equal probability of being selected. (2) focal loss (Focal) [20]: This lowers the contribution of head samples and increases the weights of tail samples; (3) label-distribution-aware margin loss (LDAM) [2]: This regularizes tail samples more strongly than the head classes; (4) re-weighting (RW) [13]: Each class is attached to weight by inverse class frequency; (5) class-balanced re-weighting (CB) [6]: A variant of RW depending on the inverse effective number in each class; (6) deferred re-weighting (DRW) and (4) deferred class-balanced re-weighting (DCB) [2]: RW and CB are delayed until the later stage of training.

Transfer learning methods. (1) M2m [17]: a variant of RS with transfer learning, which translates the head samples to tail classes; (2) BBN [48]: Transferring the re-sampling feature to the long-tailed data.

When combining one loss and one method, we concatenate them with a dash as a new technique. A state-of-the-art method, LDAM-DRW, combines the LDAM loss and the DRW method.

4.4 Evaluation measures

We evaluate the models on the corresponding test datasets. We mainly report the most popular top-1 accuracy (ACC) over all classes. To better examine performance variations across classes with the different number of samples seen during training, we further report ACC on tail classes (80% few classes). Also, we report a hierarchical evaluation measure [18]. Evaluation measures are reported as a percentage.

ACC. ACC is a commonly used evaluation measure for classification tasks. Let a label with the highest probability denote the model predicted label. ACC is calculated as a ratio of the number of correctly predicted samples to the total number of samples.

Hierarchical $\mathbf{F_1F_H}$. F_H used in hierarchical classification tasks is defined as

$$F_H \quad 2\frac{P_H R_H}{P_H \quad R_H}, \tag{13}$$

where P_H denotes hierarchical precision, and R_H denotes hierarchical recall. They are written as $P_H \quad \frac{|\ \hat{D}_{aug} \quad D_{aug}\ |}{|\ \hat{D}_{aug}\ |}$, $R_H \quad \frac{|\ \hat{D}_{aug} \quad D_{aug}\ |}{|\ D_{aug}\ |}$, where $D_{aug} \quad D \ AnD$, $\hat{D}_{aug} \quad \hat{D} \ An\hat{D}$, D denotes a true label of test sample, AnD denotes a parent node set of the true class to which the sample really belongs, $\hat{D}$ indicates a predicted label of the test sample, and $|\cdot|$ denotes a count of elements.

5 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we first describe an ablation study of the proposed model. Second, we validate and analyze the semantic gap. Third, we explore the effectiveness of different numbers of clustering super-classes. We employ CIFAR-IR50 (L 10) as an example in all cases to facilitate a fair comparison. Finally, we present the performance of our model compared with state-of-the-art methods on several datasets.

5.1 Ablation study

We conduct an ablation study to evaluate the role of each component in the HCKC loss. The HCKC loss has three design choices: a weighted semantic term, a weighted clustering term, and a transfer strategy, denoted as "Semantic", "Clustering", and "Transfer", respectively. We verify the effectiveness of each component in Table 2.

Table 2. Ablation study of each choice in HCKC loss on CIFAR-IR50. (Best ACC (%) of all tail classes is marked in bold.)

Semantic	Clustering	Transfer	All	Tail
			43.31	35.89
√			43.79	36.34
√		√	44.42	36.96
	√	√	45.12	37.59
√	√		44.59	37.10
√	√	√	**45.31**	**38.15**

Weighted semantic term. Our initial motivation is to use class similarity to aid visual classification. The weighted semantic term that mitigates the poor representation leads to a 0.48% ACC improvement for all classes. This improvement for tail classes further proves that semantic knowledge promotes tail learning. These observations suggest that semantic super-classes are desirable for long-tailed tasks.

Weighted clustering term. The clustering hierarchical relationship generates a class relationship from the data, which is different from the case with semantic relations. The weighted clustering term uses data similarity to promote the generalization of targets, for which we observe a significant improvement from 43.31% to 45.12%. Meanwhile, the ACC of tail classes is 1.70% better than that of the baseline.

Complementary strategy. Semantic gap problems always exist in semantic super-classes. We design a complementary strategy to combine two weighted terms, which simultaneously considers semantic and clustering knowledge. The combination of semantic and weighted clustering terms aims to reduce the dependence on semantic knowledge and achieves 44.59% ACC, which outperforms the method using the weighted semantic term alone by 0.8% ACC. This reveals the effectiveness of using clustering knowledge to avoid semantic gaps.

Knowledge transfer. The knowledge transfer strategy is another crucial component in our loss function. We employ the strategy to realize progressive learning between tasks, which is different from the case when using pre-trained super-class tasks (λ 1). We observe that using a combination with a transfer strategy outperforms one without a transfer strategy by 0.72% ACC; in particular, it improves the ACC of tail classes from 37.10This shows that knowledge can transfer from a super task to a target task.

The variation in validation loss for tail class accuracy across training epochs is shown in Figure 5. It illustrates the trend in the loss decrement under the regimes of the semantic training phase, clustering representation training phase, and knowledge transfer phase.

From Figure 5, we can observe that the overall trend in the loss function is that the fluctuations are decreasing. During the semantic guidance in the early stage of training, the loss function curve declines rapidly in the early stage and then fluctuates. We then utilize cluster representation to guide optimization of the loss function. In this

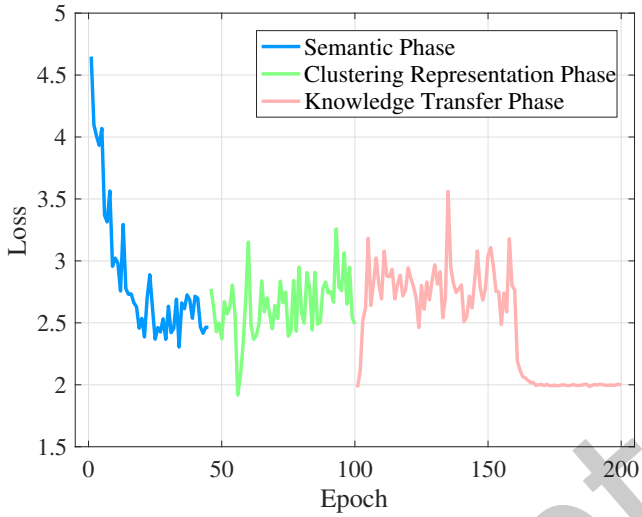


Fig. 5. Variations in loss decrement under different regimes: semantic training phase, clustering representation training phase, and knowledge transfer phase.

period, the performance curve fluctuates, reflecting the effect of cluster representation guidance. Finally, we use the knowledge transfer method to transfer clustering knowledge into semantic guidance to bridge the semantic gap problem. Therefore, the loss function presents an oscillatory decline during the knowledge transfer period and then continues declining to a point of stable convergence.

5.2 Validations for semantic gap

We leverage similar classes to improve the generalization ability of the CNN. Semantic super-classes are first used to extract the feature representations. However, semantic gaps are inevitable in classification tasks, where semantically consistent classes are not necessarily visually similar. To verify this, we cluster visually similar classes in the feature space on CIFAR-IR50. Then, we compare the differences between semantic and clustering relationships in Table 3. Moreover, we provide some example images in Figure 6 to give an intuitive interpretation of the semantic gaps . These images especially come from the corresponding experimental dataset.

We obtain the following observations from Table 3 and Figure 6:

(1) In Table 3, the semantic super-class *Natural scenes* contains the classes *Forest*, *Cloud*, *Mountain*, *Plain*, and *Sea*. The clustering super-class C_2 contains all of these classes except *Forest*, which suggests there is target inconsistency under the semantic and clustering super-classes. The class *Forest* belonging to super-class C_1 is separated from the other classes, which means that *Forest* features are dissimilar to other class features. Then, dissimilar classes are grouped into the same semantic super-class to train a generalized representation. Hence, the semantic knowledge extracted by the hierarchical CNN is deficient. This proves that clustering knowledge is used to compensate for imperfect semantic knowledge.

(2) In Table 3, the class *Forest* leads to a semantic gap under the super-class *Natural scenes*. One hypothesis regarding semantic super-classes is that semantically similar classes are likely to be visually similar. Features extracted from visually similar classes facilitate learning and generalization of the CNN. However, *Forest* causes a poor representation in the feature space. Thus, this semantic gap problem degrades classification performance.

Table 3. The target consistency under the semantic and clustering super-classes on experimental dataset LT-CIFAR-100. The semantic and clustering super-classes represent semantic and visual similarities, respectively. (Inconsistent target classes are marked in bold.)

Type	Super-class	Target
Semantic	*Natural scenes*	***Forest***, *Cloud*, *Mountain*, *Plain*, *Sea*
	Fruit & vegetables	***Mushroom***, *Apple*, *Orange*, *Pear*, *Pepper*
	...	...
Clustering	C_1	***Forest***, ...
	C_2	*Cloud*, *Mountain*, *Plain*, *Sea*,...
	C_3	***Mushroom***, ...
	C_4	*Apple*, *Orange*, *Pear*, *Pepper*, ...
	...	...

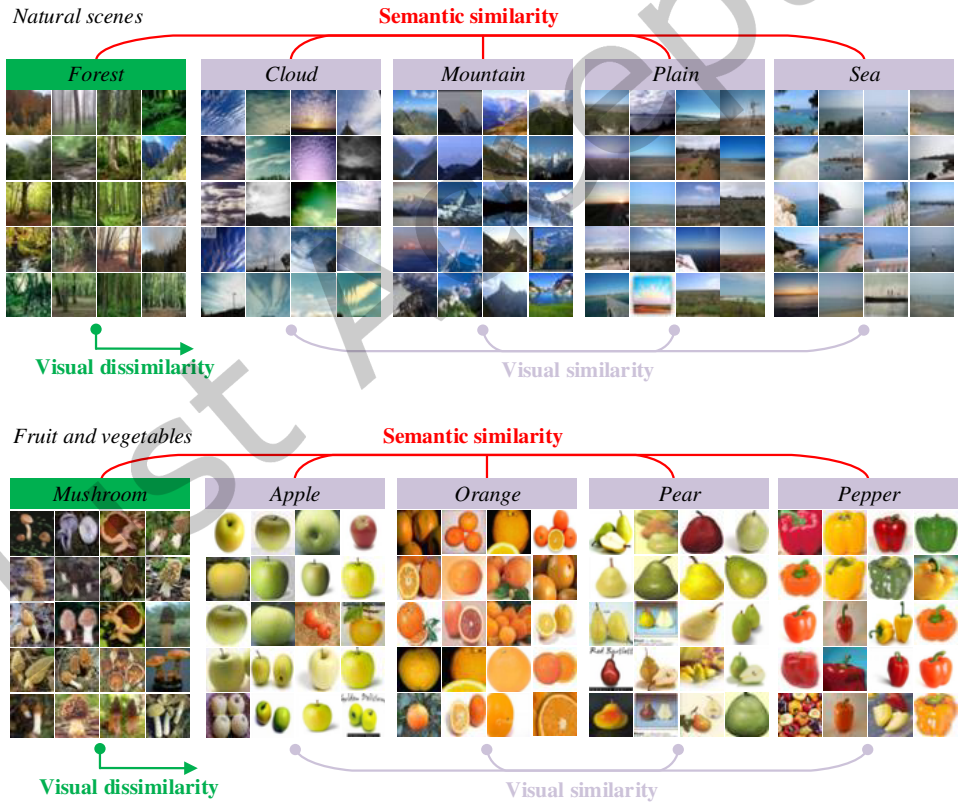


Fig. 6. Validations for a semantic gap that semantically similar classes are not necessarily visually similar. These example images come from the experimental dataset LT-CIFAR-100.

(3) In Figure 6, for the semantic super-class *Natural scenes*, class *Forest* is semantically similar but not visually similar to other classes. For example, *Forest* has the color green, while *Cloud* is blue. The texture of the *Forest* image is vertical, while that of *Plain* is horizontal. These differences in appearance are regarded as semantic similarities.

Figure 7 illustrates the t-SNE visualization of five classes in LT-CIFAR-100 with ρ 10 and ρ 50. Different colors represent the different classes, which are *Forest*, *Cloud*, *Mountain*, *Mushroom*, and *Apple*. Figure 7 demonstrates clusters that have the same semantic superclass but have been clustered differently. This leads to inconsistency between the semantic and clustering relationships. Therefore, we employ the complementary strategy to balance the two pieces of knowledge.

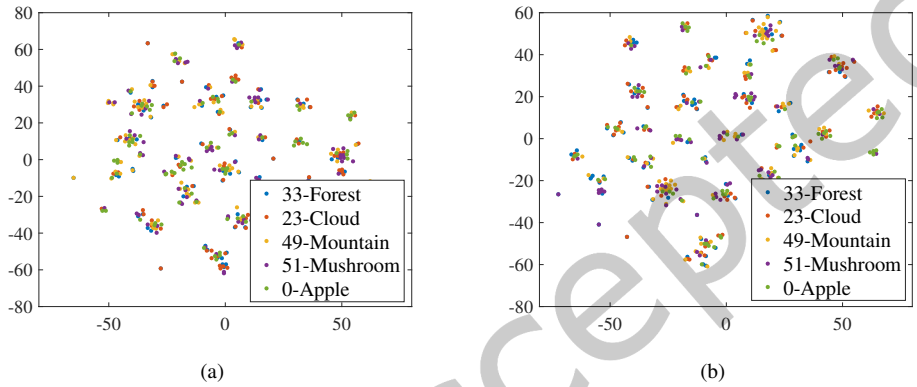


Fig. 7. t-SNE visualization of five classes in LT-CIFAR-100 with (a) ρ 10 and (b) ρ 50.

5.3 Analysis of clustering super-class number

The number of semantic super-classes is fixed according to the corresponding dataset. The number of clustering super-classes L is a hyper-parameter that influences the performance of the weighted clustering term. Hyper-parameter L comes from a fixed set of candidates; *i.e.*, $L \in \{5, 10, 15, 20, 25, 30\}$. We report the ACCs of all and tail classes and and discuss the effect of L on our model. The classification performance on CIFAR-IR50 with different values of L is shown in Figure 8.

From Figure 8, we obtain the following observations:

(1) Different values of L results in different ACCs, meaning that different datasets have various hyperparameters.

(2) Consistent trends in ACC between all and tail classes suggest that our model is appropriate for long-tailed classification. The best L is 10. The ACC is best when L = 10. The performance is consistent between all classes and tail classes.

(3) When the number of clustering super-classes is too small, the clustering knowledge representation is less helpful for learning target classes. When the number of clustering super-classes is too large, it makes the clustering features similar to those of the targets. This conflicts with our aim for generalized knowledge to contribute to tail classification.

(4) Although several relatively unreasonable L values negatively impact the training of our model, their performance exceeds the baseline. For example, when L 5, the ACC is improved from 43.31% to 44.10%, demonstrating the proposed model's capability to handle long-tailed challenges.

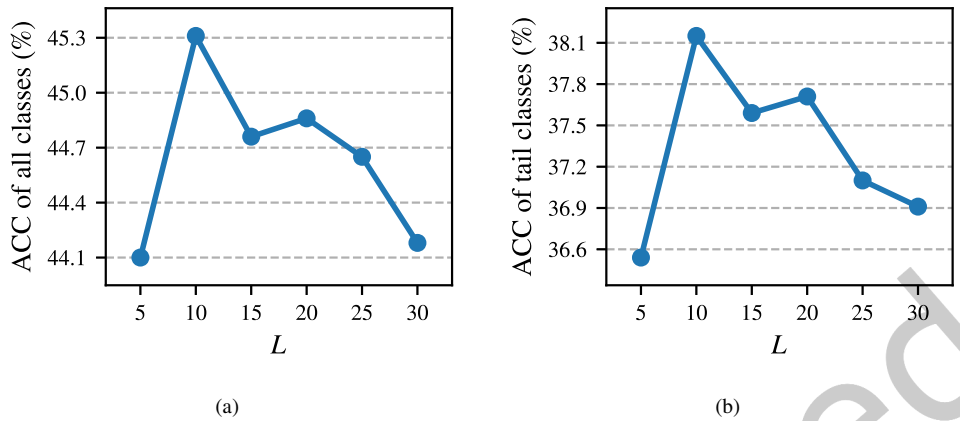


Fig. 8. ACCs of all classes and tail classes according to the number of different clustering super-classes on CIFAR-IR50.

5.4 Experimental results on artificial datasets

We conduct extensive experiments on four artificial datasets: LT-CIFAR-10, LT-TYIN, LT-CIFAR-100, and LT-TDIN. LT-CIFAR contains three imbalanced versions with imbalanced ratios of 10, 50, and 100. LT-TYIN contains three imbalanced versions with imbalanced ratios of 10, 100, and 500. LT-TDIN contains one imbalanced version with an imbalanced ratio of 200. We compare our model with three groups of methods: re-sampling, re-weighting, and transfer learning. We reproduce most of them from their open codes for a fair comparison. All of these experiments are conducted under the same experimental settings. We set L 10 as the number of clustering super-classes for their datasets. Table 4 reports the ACCs of the various methods on LT-CIFAR-10 and LT-TYIN.

Table 4. Comparison of classification performance on LT-CIFAR-10 and LT-TYIN. (Best ACC (%) is marked in bold, and † denotes a copied result from its author.)

Dataset	LT-CIFAR-10			LT-TYIN		
Imbalance ratio	ρ 10	ρ 100	ρ 500	ρ 10	ρ 100	ρ 500
CE	86.03	69.75	61.72	53.15	38.72	31.77
SMOTE [4]	86.12	71.12	60.65	53.32	38.04	30.26
CB [6]	86.26	72.85	62.39	53.64	39.14	31.52
Focal [20]	86.45	73.90	63.68	54.10	39.76	32.06
LDAM [2]	86.97	75.01	62.37	54.89	41.51	32.83
M2m [17]	87.31	74.85	64.59	55.21	41.72	32.67
Equalization† [35]	87.08	72.61	58.29	52.75	37.39	29.63
Seesaw† [38]	87.76	75.18	64.48	54.60	40.98	34.07
GHM† [45]	87.29	75.39	**66.13**	55.35	42.66	34.79
HCKC (ours)	**87.81**	**77.05**	65.90	**56.24**	**43.32**	**35.88**

The experimental results show that:

(1) HCKC exceeds CE when used as the baseline method on LT-CIFAR-10 and LT-TYIN. For example, when ρ 500, HCKC enhances the accuracy (ACC) by 4.18% and 4.11% on the LT-CIFAR-10 and LT-TYIN datasets,

respectively. This result underscores the value of the hierarchical class structure as a tool for addressing long-tailed classification challenges.

(2) HCKC outperforms Focal and LDAM on both LT-CIFAR-10 and LT-TYIN; for example, when ρ 100 on LT-CIFAR-10, HCKC boosts ACC by 3.15% and 2.04%, respectively, highlighting the comparable effectiveness of hierarchical knowledge in comparison to the other methods. This result underscores the efficacy of the proposed model.

(3) Our approach consistently outperforms the m2m method across both datasets, which are characterized by various degrees of class imbalance. For instance, on LT-CIFAR-10 with an imbalance ratio of 100, HCKC achieves 2.2% better accuracy than m2m Similarly, when = 500 on LT-TYIN, HCKC demonstrates a 3.21% accuracy improvement over m2m. These experimental outcomes underscore the robustness and effectiveness of HCKC, even in scenarios of substantial dataset imbalance. It is evident that HCKC maintains its superior performance under extremely imbalanced conditions.

(4) HCKC surpasses most of the existing methods. The enhancements observed in our model underscore the detrimental effects of redundant features on representation learning. This highlights the importance of similar features and correlated knowledge in promoting network generalization.

Table 5 reports the ACCs of various methods on LT-CIFAR-10 and LT-TYIN. We obtain the following observations:

(1) HCKC outperforms CE across all datasets. For example, HCKC improves ACC by 2% on CIFAR-IR50, which means that the hierarchical class structure is an effective tool for long-tailed classification. This further reveals that similar features benefit the training of the network.

(2) HCKC exceeds Focal and LDAM across all datasets except CIFAR-IR100, demonstrating that hierarchical knowledge is equally effective compared with other methods. It suggests the effectiveness of the proposed model.

(3) HCKC outperforms most of the existing methods. Re-sampling or re-weighting the data distribution alleviates imbalanced training, which avoids making a biased classifier. However, this brings about tail data overlap

Table 5. Comparison of classification performance on LT-CIFAR-100 and LT-TDIN. (Best ACC (%) is marked in bold, and † denotes a copied result from its author.)

Dataset	LT-CIFAR-100			LT-TDIN
Imbalance ratio	$\rho \approx 10$	$\rho \approx 50$	ρ 100	$\rho \approx 200$
CE-RS [16]	55.93	40.71	35.36	12.38
CE-DRS [2]	57.00	44.06	38.61	14.46
SMOTE† [4]	53.80	-	34.00	-
CE	56.52	43.31	38.34	15.11
CE-RW [13]	55.92	42.01	32.41	10.29
CE-CB [6]	56.26	40.12	33.15	10.29
Focal [20]	55.68	45.06	39.62	15.33
Focal-DCB [20]	57.46	45.87	38.54	12.52
LDAM [2]	57.03	45.21	39.98	14.80
LDAM-DRW [2]	58.10	45.42	41.38	11.33
M2m† [17]	57.60	-	-	-
BBN† [48]	59.12	47.02	-	-
HCKC (ours)	57.56	45.31	39.00	15.45
HCKC-DRW (ours)	**59.45**	**47.78**	**41.76**	**18.79**

and information repetition. For example, CE-RS and CE-RW performed worse than CE by 2.98% and 5.93%, respectively, which implies that information overlap causes over-fitting of tail classes and degradation of network training. The improvements of our model prove that repeated features hurt representation learning. This indicates that similar features and correlative knowledge are helpful for network generalization.

(4) HCKC is inferior to the transfer learning methods M2m and BBN. Although transfer learning introduces irrelevant head knowledge to tail classes, it enriches tail data and increases tail information, alleviating a data-hungry representation learning limitation. This proves that the success of deep learning is inseparable from the availability of large-scale datasets. Compared with transfer learning, our model focuses on similar information and related knowledge, ignoring the effect of large-scale datasets on the deep network. Thus, we further combine HCKC and DRW to ease the problems of class imbalance and missing samples and exhaustively improve the classification performance of long-tailed tasks.

(5) HCKC-DRW outperforms all compared methods across all datasets, which indicates that we need to use relevant knowledge to facilitate long-tailed classification. In particular, the ACC of HCKC-DRW is 3.46% better than that of Focal on LT-TDIN.

5.5 Experimental results on real-world datasets

We present the results of two evaluation measures on the SUN-324 and iNaturalist datasets to demonstrate the effectiveness of the proposed model on real-world datasets. According to the results for artificial datasets, dissimilar classes are clustered into a super-class when the number of clustering super-classes is too small. On the contrary, the super-class task provides information that is too specific when the number of clustering super-classes is too large, which makes the clustering features similar to those of the targets. SUN-324 and iNaturalist have 15 and 13 semantic super-classes, respectively. Therefore, we let the number of clustering super-classes L be 5 for SUN-324 and iNaturalist. Table 6 reports the ACC and F_H values of the comparison methods.

Table 6. Comparison of classification performance on real-world datasets. (Best results (%) are marked in bold.)

Dataset	SUN-324		iNaturalist	
Imbalance ratio	$\rho \approx 27.9$		$\rho \approx 435.4$	
Evaluation measure	ACC	F_H	ACC	F_H
CE-RS [16]	37.48	60.95	8.04	46.52
CE-DRS [2]	38.52	61.66	8.33	46.63
CE	35.03	60.08	7.08	46.24
CE-RW [13]	37.83	61.16	6.37	45.49
CE-CB [6]	38.23	61.50	6.27	45.42
Focal [20]	35.24	60.48	7.26	46.29
Focal-DCB [20]	37.91	61.21	6.45	45.47
LDAM [2]	31.46	57.97	7.96	46.72
LDAM-DRW [2]	35.46	59.70	7.26	46.01
HCKC (ours)	35.51	60.69	7.19	46.30
HCKC-DRW (ours)	**39.95**	**62.48**	**9.01**	**47.03**

From Table 6, we obtain the following observations:

(1) HCKC exceeds CE (used as the baseline method), which implies that the proposed model is appropriate for real-world data distributions.

(2) For SUN-324, HCKC-DRW consistently obtains better performance than all compared methods by a large margin (at least 1.43%). In particular, the ACCs of HCKC-DRW are 1.43% and 8.49% better on CE-DRS and LDAM, respectively.

(3) For iNaturalist, HCKC-DRW outperforms all compared methods, indicating the effectiveness of our model in cases with many classes and extreme class imbalance. This improves ACC by 0.68% and 1.93% compared with CE-DRS and CE, respectively.

(4) We obtain observations similar to ACC from the perspective of F_H. Our model achieves the best performance according to ACC and F_H, which reveals that the proposed model promotes the training of a robust network.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a hierarchical CNN with a knowledge complementary strategy for long-tailed classification. First, we constructed a hierarchical CNN to exploit auxiliary information and pay attention to tail classes. Then, we designed two hierarchical weighted terms, a weighted semantic term, and a weighted clustering term, to balance the knowledge extracted by the complementary strategy. The weighted semantic term uses semantic knowledge to promote the generalization of tail classes. Moreover, the weighted clustering term learns clustering knowledge and reduces the dependence of the CNN on semantic knowledge. Extensive experiments show that HCKC achieves satisfactory performance on long-tailed datasets. It demonstrates that using hierarchical knowledge as auxiliary information can alleviate under-fitting of target classes due to insufficient tail data. In HCKC, we only utilize the super-class information to facilitate tail-class learning. In the future, from the knowledge transfer perspective, we will consider how to transfer head class knowledge to the tail class and design a technique to combine the super-class and head information. From the model design perspective, we will also consider further integration of pre-trained models for supervised or self-supervised learning to investigate the impact of feature fusion on classification outcomes.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant No. 62376114 and the Natural Science Foundation of Fujian Province under Grant Nos. 2021J011003 and 2021J011004.

REFERENCES

[1] Lida Abdi and Sattar Hashemi. 2015. To combat multi-class imbalanced problems by means of over-sampling techniques. *Transactions on Knowledge and Data Engineering* 28, 1 (2015), 238–251.

[2] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. 2019. Learning imbalanced datasets with label-distribution-aware margin loss. In *International Conference on Neural Information Processing Systems*. 1567–1578.

[3] Jianlong Chang, Gaofeng Meng, Lingfeng Wang, Shiming Xiang, and Chunhong Pan. 2018. Deep self-evolution clustering. *Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2018), 809–823.

[4] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Artificial Intelligence Research* 16 (2002), 321–357.

[5] Haibin Chen, Qianli Ma, Zhenxi Lin, and Jiangyue Yan. 2021. Hierarchy-aware label semantics matching network for hierarchical text classification. In *Association for Computational Linguistics*. 4370–4379.

[6] Yin Cui, Menglin Jia, Tsung Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *Conference on Computer Vision and Pattern Recognition*. 9268–9277.

[7] Jia Deng, Wei Dong, Richard Socher, Li Jia Li, Kai Li, and Li Fei Fei. 2009. ImageNet: a large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*. 248–255.

[8] Jia Deng, Jonathan Krause, Alexander C Berg, and Li Fei Fei. 2012. Hedging your bets: optimizing accuracy-specificity trade-offs in large scale visual recognition. In *Conference on Computer Vision and Pattern Recognition*. 3450–3457.

[9] Saite Fan, Xinmin Zhang, and Zhihuan Song. 2022. Imbalanced sample selection with deep reinforcement learning for fault diagnosis. *Transactions on Industrial Informatics* 18, 4 (2022), 2518–2527.

[10] Yubin Ge, Site Li, Xuyang Li, Fangfang Fan, Wanqing Xie, Jane You, and Xiaofeng Liu. 2021. Embedding semantic hierarchy in discrete optimal transport for risk minimization. In *International Conference on Acoustics, Speech and Signal Processing*. 2835–2839.

[11] Hao Guo and Song Wang. 2021. Long-tailed multi-label visual recognition by collaborative training on uniform and re-balanced samplings. In *Conference on Computer Vision and Pattern Recognition*. 15089–15098.
[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*. 770–778.
[13] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. 2016. Learning deep representation for imbalanced classification. In *Conference on Computer Vision and Pattern Recognition*. 5375–5384.
[14] Ling Chien Hung, Ya Han Hu, Chih Fong Tsai, and Min Wei Huang. 2022. A dynamic time warping approach for handling class imbalanced medical datasets with missing values: a case study of protein localization site prediction. *Expert Systems with Applications* 192 (2022), 116437.
[15] Matheus Inoue, Carlos Henrique Forster, and Antonio Carlos dos Santos. 2020. Semantic hierarchy-based convolutional neural networks for image classification. In *International Joint Conference on Neural Networks*. 1–8.
[16] Nathalie Japkowicz. 2000. The class imbalance problem: significance and strategies. In *International Conference on Artificial Intelligence*. 111–117.
[17] Jaehyung Kim, Jongheon Jeong, and Jinwoo Shin. 2020. M2m: imbalanced classification via major-to-minor translation. In *Conference on Computer Vision and Pattern Recognition*. 13896–13905.
[18] Aris Kosmopoulos, Ioannis Partalas, Eric Gaussier, Georgios Paliouras, and Ion Androutsopoulos. 2015. Evaluation measures for hierarchical classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery* 29, 3 (2015), 820–865.
[19] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. *University of Tront* (2009), 1–58.
[20] Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollr. 2017. Focal loss for dense object detection. In *International Conference on Computer Vision*. 2980–2988.
[21] Wei Chao Lin, Chih Fong Tsai, Ya Han Hu, and Jing Shang Jhang. 2017. Clustering-based undersampling in class-imbalanced data. *Information Sciences* 409 (2017), 17–26.
[22] Huafeng Liu, Jiaqi Wang, and Liping Jing. 2021. Cluster-wise hierarchical generative model for deep amortized clustering. In *Conference on Computer Vision and Pattern Recognition*. 15109–15118.
[23] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. 2019. Large-scale long-tailed recognition in an open world. In *Conference on Computer Vision and Pattern Recognition*. 2537–2546.
[24] Jianghong Ma, Tommy WS Chow, and Haijun Zhang. 2022. Semantic-gap-oriented feature selection and classifier construction in multilabel learning. *Transactions on Cybernetics* 52, 1 (2022), 101–115.
[25] Sebastin Maldonado, Carla Vairetti, Alberto Fernandez, and Francisco Herrera. 2022. FW-SMOTE: a feature-weighted oversampling approach for imbalanced classification. *Pattern Recognition* 124 (2022), 108511.
[26] Stanislav Naumov, Grigory Yaroslavtsev, and Dmitrii Avdiukhin. 2021. Objective-based hierarchical clustering of deep embedding vectors. In *Conference on Artificial Intelligence*. 9055–9063.
[27] Abraham Montoya Obeso, Jenny Benois-Pineau, Mireya Sara Garca Vzquez, and Alejandro lvaro Ramrez Acosta. 2022. Visual vs internal attention mechanisms in deep neural networks for image classification and object detection. *Pattern Recognition* 123 (2022), 108411.
[28] Peter Oram. 2001. WordNet: an electronic lexical database. *Applied Psycholinguistics* 22, 1 (2001), 131–134.
[29] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. 2018. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*.
[30] Yong Rui, Thomas S Huang, and Shih Fu Chang. 1999. Image retrieval: current techniques, promising directions, and open issues. *Journal of Visual Cmmunication and Image Representation* 10, 1 (1999), 39–62.
[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.
[32] Sungho Suh, Paul Lukowicz, and Yong Oh Lee. 2022. Discriminative feature generation for classification of imbalanced data. *Pattern Recognition* 122 (2022), 108302.
[33] Muhammad Atif Tahir, Josef Kittler, and Fei Yan. 2012. Inverse random under sampling for class imbalance problem and its application to multi-label classification. *Pattern Recognition* 45, 10 (2012), 3738–3750.
[34] Jingru Tan, Xin Lu, Gang Zhang, Changqing Yin, and Quanquan Li. 2021. Equalization loss v2: a new gradient balance approach for long-tailed object detection. In *Conference on Computer Vision and Pattern Recognition*. 1685–1694.
[35] Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. 2020. Equalization loss for long-tailed object recognition. In *Conference on Computer Vision and Pattern Recognition*. 11662–11671.
[36] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. 2018. The iNaturalist species classification and detection dataset. In *Conference on Computer Vision and Pattern Recognition*. 8769–8778.
[37] Guoyin Wang, Jie Yang, and Ji Xu. 2017. Granular computing: from granularity optimization to multi-granularity joint problem solving. *Granular Computing* 2, 3 (2017), 105–120.

[38] Jiaqi Wang, Wenwei Zhang, Yuhang Zang, Yuhang Cao, Jiangmiao Pang, Tao Gong, Kai Chen, Ziwei Liu, Chen Change Loy, and Dahua Lin. 2021. Seesaw loss for long-tailed instance segmentation. In *Conference on Computer Vision and Pattern Recognition*. 9695–9704.
[39] Yu Wang, Ruonan Liu, Di Lin, Dongyue Chen, Ping Li, Qinghua Hu, and C. L. Philip Chen. 2021. Coarse-to-fine: progressive knowledge transfer based multi-task convolutional neural network for intelligent large-scale fault diagnosis. *Transactions on Neural Networks and Learning Systems* (2021), 1–14.
[40] Yu Xiong Wang, Deva Ramanan, and Martial Hebert. 2017. Learning to model the tail. In *Conference on Neural Information Processing Systems*. 7032–7042.
[41] Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. 2016. SUN database: exploring a large collection of scene categories. *International Journal of Computer Vision* 119, 1 (2016), 3–22.
[42] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. 2010. SUN database: large-scale scene recognition from abbey to zoo. In *Conference on Computer Vision and Pattern Recognition*. 3485–3492.
[43] Chaoyang Xu, Renjie Lin, Jinyu Cai, and Shiping Wang. 2022. Deep image clustering by fusing contrastive learning and neighbor relation mining. *Knowledge-Based Systems* 238 (2022), 107967.
[44] Huaikuan Yi, Qingchao Jiang, Xuefeng Yan, and Bei Wang. 2021. Imbalanced classification based on minority clustering synthetic minority oversampling technique with wind turbine fault detection application. *Transactions on Industrial Informatics* 17, 9 (2021), 5867–5875.
[45] Renhui Zhang, Tiancheng Lin, Rui Zhang, and Yi Xu. 2022. Solving the long-tailed problem via intra-and inter-category balance. In *International Conference on Acoustics, Speech and Signal Processing*. 2355–2359.
[46] Hong Zhao, Qinghua Hu, Pengfei Zhu, Yu Wang, and Ping Wang. 2021. A recursive regularization based feature selection framework for hierarchical classification. *Transactions on Knowledge and Data Engineering* 33, 7 (2021), 2833–2846.
[47] Wei Zhong and Feng Gu. 2020. Predicting local protein 3D structures using clustering deep recurrent neural network. *Transactions on Computational Biology and Bioinformatics* (2020).
[48] Boyan Zhou, Quan Cui, Xiu Shen Wei, and Zhao Min Chen. 2020. BBN: bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Conference on Computer Vision and Pattern Recognition*. 9719–9728.
[49] Ning Zhou and Jianping Fan. 2013. Jointly learning visually correlated dictionaries for large-scale visual recognition applications. *Transactions on Pattern Analysis and Machine Intelligence* 36, 4 (2013), 715–730.
[50] Yu Zhou, Xiaoni Li, Yucan Zhou, Yu Wang, Qinghua Hu, and Weiping Wang. 2022. Deep collaborative multi-task network: a human decision process inspired model for hierarchical image classification. *Pattern Recognition* 124 (2022), 108449.
[51] Linchao Zhu and Yi Yang. 2022. Label independent memory for semi-supervised few-shot video classification. *Transactions on Pattern Analysis and Machine Intelligence* 44, 1 (2022), 273–285.