Solving Long-Tailed Recognition with Deep Realistic Taxonomic Classifier

Tz-Ying Wu(✉), Pedro Morgado, Pei Wang, Chih-Hui Ho, and Nuno Vasconcelos

University of California, San Diego, USA
{tzw001,pmaravil,pew062,chh279,nvasconcelos}@ucsd.edu

Abstract. Long-tail recognition tackles the natural non-uniformly distributed data in real-world scenarios. While modern classifiers perform well on populated classes, its performance degrades significantly on tail classes. Humans, however, are less affected by this since, when confronted with uncertain examples, they simply opt to provide coarser predictions. Motivated by this, a *deep realistic taxonomic classifier* (Deep-RTC) is proposed as a new solution to the long-tail problem, combining realism with hierarchical predictions. The model has the option to reject classifying samples at different levels of the taxonomy, once it cannot guarantee the desired performance. Deep-RTC is implemented with a stochastic tree sampling during training to simulate all possible classification conditions at finer or coarser levels and a rejection mechanism at inference time. Experiments on the long-tailed version of four datasets, CIFAR100, AWA2, Imagenet, and iNaturalist, demonstrate that the proposed approach preserves more information on all classes with different popularity levels. Deep-RTC also outperforms the state-of-the-art methods in long-tailed recognition, hierarchical classification, and learning with rejection literature using the proposed *correctly predicted bits* (CPB) metric.

Keywords: Realistic predictor · Taxonomic classifier · Long-tail recognition

1 Introduction

Recent advances in computer vision can be attributed to large datasets [16] and deep convolutional neural networks (CNN) [32,42,53]. While these models have achieved great success on balanced datasets, with approximately the same number of images per class, real world data tends to be highly imbalanced, with a very long-tailed class distribution. In this case, classes are frequently split into many-shot, medium-shot and few-shot, based on the number of examples [46]. Since deep CNNs tend to overfit in the small data regime, they frequently underperform for medium and few-shot classes. Popular attempts to overcome this

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58598-3_11) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12353, pp. 171–189, 2020.
https://doi.org/10.1007/978-3-030-58598-3_11

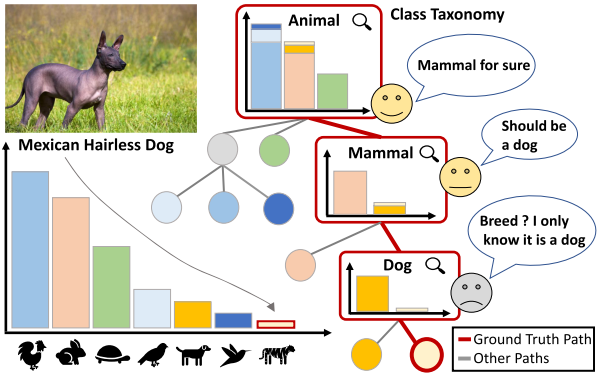

Fig. 1. Real-world datasets have class imbalance and long tails (left). Humans deal with these problems by combining class taxonomies and self-awareness (right). When faced with rare objects, like a "Mexican Hairless Dog", they push the decision to a coarser taxonomic level, e.g., simply recognizing a "Dog", of which they feel confident. This is denoted as *realistic taxonomic classification* to guarantee that all samples are processed with a high level of confidence.

limitation include data resampling [6,8,20,30], cost-sensitive losses [14], knowledge transfer from high to low population classes [46,60], normalization [38], or margin-based methods [7]. All these approaches seek to improve the classification performance of the standard softmax CNN architecture.

There is, however, little evidence that this architecture is optimally suited to deal with long-tailed recognition. For example, humans do not use this model. Rather than striving for discrimination between all objects in the world, they adopt class *taxonomies* [4,5,36,37,56], where classes are organized hierarchically at different levels of granularity, e.g. ranging from coarse domains to fine-grained 'species' or 'breeds,' as shown in Fig. 1. Classification with taxonomies is broadly denoted as *hierarchical*. The standard softmax, also known as the *flat,* classifier is a hierarchical classifier of a single taxonomic level. The use of deeper taxonomies has been shown advantageous for classification by allowing feature sharing [2, 27,39,45,61,64] and information transfer across classes [17,48,51,52,63]. While most previous works on either flat or hierarchical classification attempt to classify all images at the leaves of the taxonomic tree, independently of how difficult this is, the introduction of a taxonomy enables alternate strategies.

In this work, we explore a strategy inspired by human cognition and suited for long-tailed recognition. When humans feel insufficiently trained to answer a question at a certain level of granularity, they simply provide an answer to a coarser level, for which they feel *confident.* For example, most people do not recognize the animal of Fig. 1 as a "Mexican Hairless Dog". Instead, they change the problem from classifying dog breeds into classifying mammals and simply say it is a "Dog". Hence, a long-tailed recognition strategy more consistent with human cognition is to adopt hierarchical classification and allow decisions at *intermediate* tree levels, to achieve two goals: 1) classify all examples with high

confidence, and 2) classify each example as deep in the tree as possible without violating the first goal. Since examples from low-shot classes are harder to classify confidently than those of popular classes, they tend to be classified at earlier tree levels. This can be seen as a soft version of *realistic classification* [13,57] where a classifier refuses to process examples of low-classification confidence and is denoted *realistic taxonomic classification* (RTC). The taxonomic extension enables multiple "exit levels" for the classification, at different taxonomic levels.

RTC recognizes that, while classification at the leaves uncovers full label information, *partial* label information can still be recovered when this is not feasible, by performing the classification at intermediate taxonomic stages. The goal is then to maximize the *average* information recovered per sample, favoring correct decisions of intermediate level over incorrect decisions at the leaves. We introduce a new measure of classifier performance, denoted *correctly predicted bits* (CPB), to capture this average information and propose it as a new performance measure for long-tailed recognition. Rather than simply optimizing classification accuracy at the leaves, high CPB scores require learning algorithms that produce calibrated estimates of class probabilities at *all* tree levels. This is critical to enable accurate determination of when examples should leave the tree. For long-tailed recognition, where different images can be classified at different taxonomic levels, this calibration is particularly challenging.

We address this problem with two new contributions to the training of deep CNNs for RTC. The first is a new regularization procedure based on *stochastic tree sampling,* (STS) which allows the consideration of all possible cuts of the taxonomic tree during training. RTC is then trained with a procedure similar to dropout [55], which considers the CNNs consistent with all these cuts. The second contribution addresses the challenge that RTC requires a *dynamic* CNN, capable of generating predictions at different taxonomic levels for each input example. This is addressed with a novel *dynamic predictor synthesis* procedure inspired by parameter inheritance, a regularization strategy commonly used in hierarchical classification [51,52]. To the best of our knowledge, these contributions enable the first implementation of RTC with deep CNNs and dynamic predictors. This is denoted as *Deep-RTC*, which achieves leaf classification accuracy comparable to state of the art long-tail recognition methods, but recovers much more average label information per sample.

Overall, the paper makes three contributions. 1) we propose RTC as a new solution to the long-tailed problem. 2) the *Deep-RTC* architecture, which implements a combination of stochastic taxonomic regularization and dynamic taxonomic prediction, for implementation of RTC with deep CNNs. 3) an alternative setup for the evaluation of long-tailed recognition, based on CPB scores, that accounts for the amount of information in class predictions.

2 Related Work

This work is related to several previously explored topics.

Long-Tailed Recognition: Several strategies have been proposed to address class unbalance in recognition. One possibility is to perform data resampling [31],

by undersampling head and oversampling tail classes [6,8,20,30]. Sample synthesis [29,65] has also been proposed to increase the population of tail classes. Unlike Deep-RTC, these methods do not seek improved classification architectures for long-tailed recognition. An alternative is to transfer knowledge from head to tail classes. Inspired by meta-learning [58,59], these methods learn how to leverage knowledge from head classes to improve the generalization of tail classes [60]. [46] introduces memory features that encapsulate knowledge from head classes and uses an attention mechanism to discriminate between head and tail classes. This has some similarity with Deep-RTC, which also transfers knowledge from head to tail classes, but does so by leveraging hierarchical relations between them. Long-tailed recognition has also been addressed with cost-sensitive losses, which assign different weights to different samples. A typical approach is to weight classes by their frequency [34,47] or treat tail classes as hard examples [19]. [14] proposed a class balanced loss that can be directly applied to a softmax layer and focal loss [44]. These approaches can underperform for very low-frequency classes. [7] addressed this problem by enforcing large margins for few-shot classes, where the margin is inversely proportional to the number of class samples. While effective losses for long-tailed recognition are a goal of this work, we seek losses for calibration of taxonomic classifiers, which cost-sensitive losses do not address. Finally, inspired by the correlation between the weight norm of a class and its number of samples, [38] proposed to adjust the former after classifier training. All these approaches use the flat softmax classifier architecture and do not address the design of RTC.

Hierarchical Classification: Hierarchical classification has received substantial attention in computer vision. For example, sharing information across classes has been used for object recognition on large and unbalanced datasets [17,48,63], and defining a common hierarchical semantic space across classes has been explored for zero-shot learning [3,50]. Some of the ideas used in this work, e.g. parameter inheritance, are from this literature [18,51,52]. However, most of them precede deep learning and cannot be directly applied to modern CNNs. More recently, the ideas of sharing parameters or features hierarchically have inspired the design of CNN architectures [2,27,39,45,61,64]. Some of these do not support class taxonomies, e.g. learning hierarchical feature structures for flat classification [39,50]. Others are only applicable to a somewhat rigid two-level hierarchy [2,61]. Closer to this work are architectures that complement a flat classifier with convolutional branches that regularize its features to enforce hierarchical structure [27,45,64]. These branches can be based on hierarchies of feature pooling operators [27], or classification layers [45,64] supervised with labels from intermediate taxonomic levels. However, the use of additional layers makes the comparison to flat classifier unfair, which would undermine an important goal of the paper: to investigate the benefit of hierarchical (over flat) classification for long-tailed recognition. Hence, we avoid hierarchical architectures that add parameters to the backbone network. These methods also fail to address a central challenge of RTC, namely the need for simultaneous optimization with respect to many label sets, associated with the different levels of

the class taxonomy. This requires a dynamic network, whose architecture can change on-the-fly to enable 1) the use of different label sets to classify different samples, and 2) optimization with respect to many label sets.

Learning with Rejection. The idea of learning with rejection dates back to at least [9]. Subsequent works derive theoretical results on the error-rejection trade-off [10,21], and explore alternative rejection criteria that avoid computation of class posterior probabilities [12,13,22]. Since the introduction of deep learning has made the estimation of the posterior distribution central to classification, most recent rejection functions consist of thresholding posteriors or derived quantities, such as the posterior entropy [24,25,57]. Alternative rejection methods have also been proposed, including the use of relative distances between samples [35], Monte-Carlo dropout [23], or classification model with a routing or rejection network [11,25,57]. We adopt the simple threshold based rejection rule of [24,25,57] in our implementation of RTC. However, rejection is applied to each level of a hierarchical classifier, instead of once for a flat classifier. This resembles the hedge your bets strategy of [15,18], in that it aims to maximize the average label information recovered per sample. However, while [15,18] accumulate the class probabilities of a flat classifier, our Deep-RTC addresses the calibration of probabilities *throughout* the tree. Our experiments show that this significantly outperforms the accumulation of flat classifier probabilities. [15] further calibrates class probabilities before rejection, but calibration is only conducted a posteriori (at test time). Instead, we propose STS for training hierarchical classifiers whose predictions are inherently calibrated at all taxonomic levels.

3 Long-Tailed Recognition and RTC

This section motivates the need for RTC as a solution to long-tailed recognition.

Long-Tailed Recognition. Existing approaches formulate long-tailed recognition as flat classification, solved by some variant of the softmax classifier. This combines a feature extractor $h(\mathbf{x}; \mathbf{\Phi}) \in \mathbb{R}^k$, implemented by a CNN of parameters $\mathbf{\Phi}$, and a softmax regression layer composed by a linear transformation $\mathbf{W}$ and a softmax function $\sigma(\cdot)$

$$f(\mathbf{x}; \mathbf{W}, \mathbf{\Phi}) = \sigma(z(\mathbf{x}; \mathbf{W}, \mathbf{\Phi})) \qquad z(\mathbf{x}; \mathbf{W}, \mathbf{\Phi}) = \mathbf{W}^T h(\mathbf{x}; \mathbf{\Phi}). \tag{1}$$

These networks are trained to minimize classification errors. Since samples are limited for mid and low-shot classes, performance can be weak. Long-tailed recognition approaches address the problem with example resampling, cost-sensitive losses, parameter sharing across classes, or post-processing. These strategies are not free of drawbacks. For example, cost-sensitive or resampling methods face a "whack-a-mole" dilemma, where performance improvements in low-shot classes (e.g. by giving them more weight) imply decreased performance in more populated ones (less weight). They are also very different from the recognition strategies of human cognition, which relies extensively on class taxonomies.

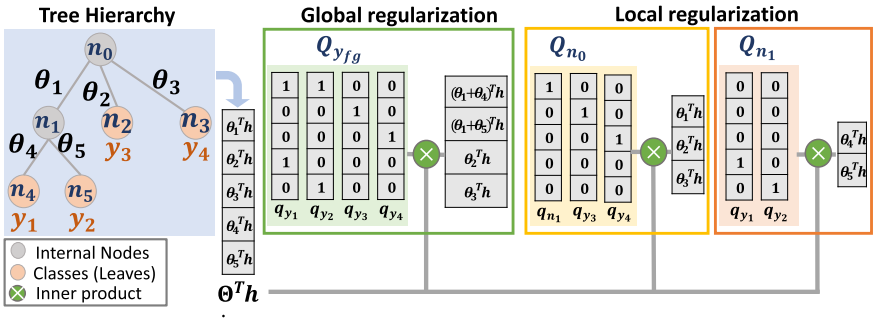

Fig. 2. Parameter sharing based on the tree hierarchy are implemented through the codeword matrices Q. The training is regularized globally from the stochastically selected label set and locally from the node-conditional consistency loss.

Many cognitive science studies have attempted to determine taxonomic levels at which humans categorize objects [4,5,36,37,56]. This has shown that most object classes have a default level, which is used by most humans to label the object (e.g. "dog" or "cat"). However, this so-called basic level is known to vary from person to person, depending on the person's training, also known as *expertise*, on the object class [4,37,56]. For example, a dog owner naturally refers to his/her pet as a "labrador" instead of as "dog." This suggests that even humans are not great long-tail recognizers. Unless they are experts (i.e. have been extensively trained in a class), they instead perform the classification at a higher taxonomic level. From a machine learning point of view, this is sensible in two ways. First, by moving up the taxonomic tree, it is always possible to find a node with sufficient training examples for accurate classification. Second, while not providing full label information for all examples, this is likely to produce a higher average label information per sample than the all-or-nothing strategy of the flat classifier [15,18]. In summary, when faced with low-shot classes, humans *trade-off classification granularity for class popularity,* choosing a classification level where their training has enough examples to guarantee accurate recognition. This does not mean that they cannot do fine-grained recognition, only that this is reserved for classes where they are experts. For example, because all humans are extensively trained on face recognition, they excel in this very fine-grained task. These observations motivate the RTC approach to long-tailed recognition.

Realistic Taxonomic Classification. A taxonomic classifier maps images $\mathbf{x} \in \mathcal{X}$ into a set of C classes $y \in \mathcal{Y} \in \{1, \ldots, C\}$, organized into a taxonomic structure where classes are recursively grouped into parent nodes according to a tree-type hierarchy $\mathcal{T}$. It is defined by a set of classification nodes $\mathcal{N} = \{n_1, \cdots, n_N\}$ and a set of taxonomic relations $\mathcal{A} = \{\mathcal{A}(n_1), \cdots, \mathcal{A}(n_N)\}$, where $\mathcal{A}(n)$ is the set of ancestor nodes of n. The finest-grained classification decisions admitted by the taxonomy occur at the leaves. We denote this set of fine-grained classes $\mathcal{Y}_{fg} = \text{Leaves}(\mathcal{T})$. Figure 2 gives an example for a classification problem with $|\mathcal{Y}_{fg}| = 4, |\mathcal{N}| = 5$, $\mathcal{A}(n_4) = \mathcal{A}(n_5) = \{n_1\}$ and

$\mathcal{A}(n_i) = \emptyset, i \in \{1, 2, 3\}$. Classes y_1, y_2 belong to parent class n_1 and the root n_0 is a dummy node containing all classes. Note that we use n to represent nodes and y to represent leaf labels. In RTC, *different samples can be classified at different hierarchy levels.* For example, a sample of class y_2 can be rejected at the root, classified at node n_1, or classified into one of the leaf classes. These options assign successively finer-grained labels to the sample. Samples rejected at the root can belong to any of the four classes, while those classified at node n_1 belong to classes y_1 or y_2. Classification at the leaves assigns the sample to a single class. Hence, RTC can predict any sub-class in the taxonomy $\mathcal{T}$. Given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{M}$ of images and class labels, and a class taxonomy $\mathcal{T}$, the *goal* is to learn a pair of classifier $f(\mathbf{x})$ and rejection function $g(\mathbf{x})$ that work together to assign each input image $\mathbf{x}$ to the finest grained class $\hat{y}$ possible, while guaranteeing certain confidence in this assignment.

The depth at which the class prediction $\hat{y}$ is made depends on the sample difficulty and the *competence-level* γ of the classification. This is a lower bound for the confidence with which $\mathbf{x}$ can be classified. A confidence score $s(f(\mathbf{x}))$ is defined for $f(\mathbf{x})$, which is declared competent (at the γ level) for classification of $\mathbf{x}$, if $s(f(\mathbf{x})) \geq \gamma$. RTC has competence level γ if all its intermediate node decisions have this competence level. While this may be impossible to guarantee for classification with the leaf label set $\mathcal{Y}_{fg}$, it can always be guaranteed by rejecting samples at intermediate nodes of the hierarchy, i.e. defining

$$g_v(\mathbf{x}; \gamma) = 1_{[s(f_v(\mathbf{x})) \geq \gamma]} \tag{2}$$

per classification node v, where $1_{[.]}$ is the Kroneker delta. This prunes the hierarchy $\mathcal{T}$ *dynamically* per sample $\mathbf{x}$, producing a customized cut $\mathcal{T}_p$ for which the hierarchical classifier is competent at a competence level γ. This pruning is illustrated on the right of Fig. 3. Samples that are hard to classify, e.g. from few-shot classes, induce low confidence scores and are rejected earlier in the hierarchy. Samples that match the classifier expertise, e.g. from highly populated classes, progress until the leaves. This is a generalization of flat realistic classifiers [57], which simply accept or reject samples. RTC mimics human behavior in that, while $\mathbf{x}$ may not be classified at the finest-grained level, confident predictions can usually be made at intermediate or coarse levels. The competence level γ offers a guarantee for the quality of these decisions. Since larger values of γ require decisions of higher confidence, they encourage sample classification early in the hierarchy, avoiding the harder decisions that are more error-prone. The trade-off between accuracy and fine-grained labeling is controlled by adjusting γ. The confidence score $s(\cdot)$ can be implemented in various ways [11,25,57]. While RTC is compatible with any of these, we adopt the popular maximum posterior probability criterion, i.e. $s(f(\mathbf{x})) = \max_i f^i(\mathbf{x})$, where $f^i(\cdot)$ is the i^{th} entry of $f(.)$. In our experience, the calibration of the node predictors $f_v(\mathbf{x})$ is more important than the particular implementation of the confidence score function.

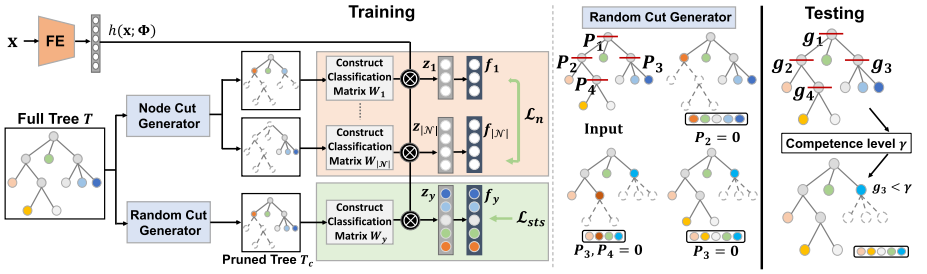

Fig. 3. Left: Deep-RTC is composed of a feature extractor, a node cut generator producing $\mathcal{Y}_n = \mathcal{C}(n)$ for all internal nodes and a random cut generator producing a potential label sets $\mathcal{Y}_c$ from $\mathcal{T}_c$. Classification matrix $\mathbf{W}_{\mathcal{Y}_c}$ is constructed for each label set and loss of (12) is imposed. Right: Rejecting samples at certain level during inference time.

4 Taxonomic Probability Calibration

In this section, we introduce the architecture of Deep-RTC.

Taxonomic Calibration. Since RTC requires decisions at all levels of the taxonomic tree, samples can be classified into any potential label set $\mathcal{Y}$ containing leaf nodes of any cut of $\mathcal{T}$. For example, the taxonomy of Fig. 2 admits two label sets, namely, $\mathcal{Y}_{fg} = \{y_1, y_2, y_3, y_4\}$ containing all classes and $\mathcal{Y} = \{n_1, y_3, y_4\}$ obtained by pruning the children of node n_1. For long-tailed recognition, where different images can be classified at very different taxonomic levels, it is important to calibrate the posterior probability distributions of *all* these label sets. We address this problem by optimizing the ensemble of *all* classifiers implementable with the hierarchy, i.e., minimize the loss

$$\mathcal{L}_{ens} = \frac{1}{|\Omega|} \sum\nolimits_{\mathcal{Y} \in \Omega} L_{\mathcal{Y}}, \tag{3}$$

where Ω is the set of all target label sets $\mathcal{Y}$ that can be derived from $\mathcal{T}$ by pruning the tree and $L_{\mathcal{Y}}$ is a loss function associated with label set $\mathcal{Y}$. While feasible for small taxonomies, this approach does not scale with taxonomy size, since the set Ω increases exponentially with $|\mathcal{T}|$. Instead, we introduce a mechanism, inspired by dropout [55], for *stochastic tree sampling (STS)* during training. At each training iteration, a random cut $\mathcal{T}_c$ of the taxonomy $\mathcal{T}$ is sampled, and the predictor $f_{\mathcal{Y}_c}(\mathbf{x}; \mathbf{W}_{\mathcal{Y}_c}, \mathbf{\Phi})$ associated with the corresponding label set $\mathcal{Y}_c$ is optimized. For this, random cuts are generated by sampling a Bernoulli random variable $P_v \sim Bernoulli(p)$ for each internal node v with a given dropout rate p. The subtree rooted at v is pruned if $P_v = 0$. Examples of these taxonomy cuts are shown in Fig. 3. The predictor $f_{\mathcal{Y}_c}$ of (1) consistent with the target label set $\mathcal{Y}_c$ associated with the cut $\mathcal{T}_c$ is then synthesized, and the loss computed as

$$\mathcal{L}_{sts} = \frac{1}{M} \sum\nolimits_{i=1}^{M} L_{\mathcal{Y}_c}(\mathbf{x}_i, y_i). \tag{4}$$

By considering different cuts at different iterations, the learning algorithm forces the hierarchical classifier to produce well calibrated decisions for all label sets.

Parameter Sharing. The procedure above requires *on-the-fly* synthesis of predictors $f_{\mathcal{Y}_c}$ for all possible label sets $\mathcal{Y}_c$ that can be derived from taxonomy $\mathcal{T}$. This implies a *dynamic* CNN architecture, where (1) changes with the sample $\mathbf{x}$. Deep-RTC is one such architecture, inspired by the fact that, for long-tailed recognition, the predictors $f_{\mathcal{Y}_c}$ should share parameters, so as to enable information transfer from head to tail classes [46,60]. This is implemented with a combination of two parameter sharing mechanisms. First, the backbone feature extractor $h(\mathbf{x}; \mathbf{\Phi})$ is shared across all label sets. Since this enables the implementation of Deep-RTC with a single network and no additional parameters, it is also critical for fair comparisons with the flat classifier. More complex hierarchical network architectures [27,45,64] would compromise these comparisons and are not investigated. Second, the predictor of (1) should reflect the hierarchical structure of each label set $\mathcal{Y}_c$. A popular implementation of this constraint, denoted *parameter inheritance (PI),* reuses parameters of ancestors nodes $\mathcal{A}(n)$ in the predictor of node n. The column vector $\mathbf{w}_n$ of $\mathbf{W}_{\mathcal{Y}}$ is then defined as

$$\mathbf{w}_n = \theta_n + \sum\nolimits_{p \in \mathcal{A}(n)} \theta_p, \quad \forall n \in \mathcal{Y} \tag{5}$$

where θ_n are non-hierarchical node parameters. This compositional structure has two advantages. First, it leverages the parameters of parent nodes (more training data) to regularize the parameters of their low-level descendants (less training data). Second, the parameter vector θ_n of node n only needs to model the residuals between n and its parent, in order to be discriminative of its siblings. In summary, low-level decisions are simultaneously simplified and robustified.

Dynamic Predictor Synthesis. Deep-RTC is a novel architecture to enable the *dynamic* synthesis of predictors $f_{\mathcal{Y}_c}$ that comply with (5). This is achieved by introducing a codeword vector $\mathbf{q}_n \in \{0,1\}^{|\mathcal{N}|}$ per node n, containing binary flags that identify the ancestors $\mathcal{A}(n)$ of n

$$\mathbf{q}_n(v) = 1_{[v \in \mathcal{A}(n) \cup \{n\}]}. \tag{6}$$

For example, in the taxonomy of Fig. 2, $\mathbf{q}_{n_1} = (1,0,0,0,0)$ since $\mathcal{A}(n_1) = \varnothing$, and $\mathbf{q}_{n_4} = (1,0,0,1,0)$ since $\mathcal{A}(n_4) = \{n_1\}$. Codeword $\mathbf{q}_n$ encodes which nodes of $\mathcal{T}$ contribute to the prediction of node n under the PI strategy, thus providing a recipe for composing predictors for any label set $\mathcal{Y}$. A matrix of node-specific parameters $\mathbf{\Theta} = [\theta_1, \ldots, \theta_{|\mathcal{N}|}]$ where $\theta_n \in \mathbb{R}^k$ for all $n \in \mathcal{N}$ is then introduced, and $\mathbf{w}_n$ can be reformulated as

$$\mathbf{w}_n = \mathbf{\Theta}\mathbf{q}_n. \tag{7}$$

The codeword vectors of all nodes $n \in \mathcal{Y}$ are then written into the columns of a codeword matrix $\mathbf{Q}_{\mathcal{Y}} \in \{0,1\}^{|\mathcal{N}| \times |\mathcal{Y}|}$, to define a predictor as in (1),

$$f_{\mathcal{Y}}(\mathbf{x}; \mathbf{\Theta}, \mathbf{\Phi}) = \sigma(z_{\mathcal{Y}}(\mathbf{x}; \mathbf{\Theta}, \mathbf{\Phi})) \qquad z_{\mathcal{Y}}(\mathbf{x}; \mathbf{\Theta}, \mathbf{\Phi}) = \mathbf{W}_{\mathcal{Y}}^T h(\mathbf{x}; \mathbf{\Phi}), \tag{8}$$

where $\mathbf{W}_{\mathcal{Y}} = \mathbf{\Theta}\mathbf{Q}_{\mathcal{Y}}$. This enables the classification of sample $\mathbf{x}$ with respect to *any* label set $\mathcal{Y}_c$ by simply making $\mathbf{Q}_{\mathcal{Y}}$ a dynamic matrix $\mathbf{Q}_{\mathcal{Y}}(\mathbf{x}) = \mathbf{Q}_{\mathcal{Y}_c}$, as illustrated in Fig. 3.

Loss Function. Deep-RTC is trained with a cross-entropy loss

$$L_{\mathcal{Y}}(\mathbf{x}_i, y_i) = -\mathbf{y}_i^T \log f_{\mathcal{Y}}(\mathbf{x}; \mathbf{\Theta}, \mathbf{\Phi}), \tag{9}$$

where $\mathbf{y}_i$ is the one-hot encoding of $y_i \in \mathcal{Y}$. When this is used in (4), the CNN is globally optimized with respect to the label set $\mathcal{Y}_c$ associated with taxonomic cut $\mathcal{T}_c$. The regularization of the many classifiers associated with different cuts of $\mathcal{T}$ is a *global* regularization, guaranteeing that all classifiers are well calibrated. Beyond this, it is also possible to calibrate the internal node-conditional decisions. Given that a sample $\mathbf{x}$ has been assigned to node n, the node-conditional decisions are *local* and determine which of the children $\mathcal{C}(n)$ the sample should be assigned to. They consider only the target label set $\mathcal{Y}_n = \mathcal{C}(n)$ defined by the children of n. For these label sets, all nodes $v \in \mathcal{C}(n)$ share the same ancestor set $\mathcal{A}_v$ and thus the second term of (5). Hence, after softmax normalization, (5) is equivalent to $\mathbf{w}_v = \theta_v$ and the node-conditional classifier $f_n(\cdot)$ reduces to

$$f_n(\mathbf{x}; \mathbf{\Theta}, \mathbf{\Phi}) = \sigma(\mathbf{Q}_n^T \mathbf{\Theta}^T h(\mathbf{x}; \mathbf{\Phi})), \tag{10}$$

where, as illustrated in Fig. 2, the codeword matrix $\mathbf{Q}_n$ contains zeros for all ancestor nodes. Internal node decisions can thus be calibrated by noting that sample $\mathbf{x}_i$ provides supervision for all node-conditional classifiers in its ground-truth ancestor path $\mathcal{A}(y_i)$. This allows the definition of a node-conditional consistency loss per node n of the form

$$\mathcal{L}_n = \frac{1}{M} \sum\nolimits_{i=1}^{M} \frac{1}{|\mathcal{A}(y_i)|} \sum\nolimits_{n \in \mathcal{A}(y_i)} L_{\mathcal{Y}_n}(\mathbf{x}_i, y_{n,i}) \tag{11}$$

where $L_{\mathcal{Y}_n}$ is the loss of (9) for the label set $\mathcal{Y}_n$ and $y_{n,i}$ the label of $\mathbf{x}_i$ for the decision at node n. Deep-RTC is trained by minimizing a combination of these local node-conditional consistency losses and the global ensemble loss of (4)

$$\mathcal{L}_{cls} = \mathcal{L}_n + \lambda \mathcal{L}_{sts}, \tag{12}$$

where λ weights the contribution of the two terms.

Performance Evaluation. Due to the universal adoption of the flat classifier, previous long-tailed recognition works equate performance to recognition accuracy. Under the taxonomic setting, this is identical to measuring leaf node accuracy $\mathbb{E}\{1_{[\hat{y}_i = y_i]}\}$ and fails to reward trade-offs between classification granularity and accuracy. In the example of Fig. 1, it only rewards the "Mexican Hairless Dog" label, making no distinction between the labels "Dog" or "Tarantula," which are both considered errors. A taxonomic alternative is to rely on hierarchical accuracy $\mathbb{E}\{1_{[\hat{y}_i \in \mathcal{A}(y_i)]}\}$ [18]. This has the limitation of rewarding "Dog" and "Mexican Hairless Dog" equally, i.e. does not encourage finer-grained decisions. In this work, we propose that a better performance measure should capture the

amount of class label information captured by the classification. While a correct classification at the leaves captures all the information, a rejection at an intermediate node can only capture partial information. To measure this, we propose to use the number of *correctly predicted bits* (CPB) by the classifier, under the assumption that each class at the leaves of the taxonomy contributes one bit of information. This is defined as

$$\text{CPB} = \frac{1}{M}\sum_{i=1}^{M} 1_{[\hat{y}_i \in \mathcal{A}(y_i)]}\left(1 - \frac{|\text{Leaves}(\mathcal{T}_{\hat{y}_i})|}{|\text{Leaves}(\mathcal{T})|}\right) \tag{13}$$

where $\mathcal{T}_{\hat{y}_i}$ is the sub-tree rooted at $\hat{y}_i$. This assigns a score of 1 to correct classification at the leaves, and smaller scores to correct classification at higher tree levels. Note that any correct prediction of intermediate level is preferred to an incorrect prediction at the leaves, but scores less than a correct prediction of finer-grain. Finally, for flat classifiers, CPB is equal to classification accuracy.

5 Experiments

This section presents the long-tailed recognition performance of Deep-RTC.

5.1 Experimental Setup

Datasets. We consider 4 datasets. **CIFAR100-LT** [14] is a long-tailed version of [40] with "imbalance factor" 0.01 (i.e. most populated class 100× larger than rarest class). **AWA2-LT** is a long-tailed version, curated by ourselves, of [41]. It contains 30 475 images from 50 animal classes and hierarchical relations extracted from WordNet [49], leading to a 7-level imbalanced tree. The training set has an imbalance factor of 0.01, the testing set is balanced. **ImageNet-LT** [46] is a long-tailed version of [16], with 1000 classes of more than 5 and less than 1280 images per class, and a balanced test set. **iNaturalist (2018)** [1,33] is a large-scale dataset of 8 142 classes with the class imbalance factor of 0.001, and a balanced test set. While the full iNaturalist dataset is used for comparisons to previous work, a more manageable subset, iNaturalist-sub, containing 55 929 images for training and 8 142 for testing, is used for ablation studies. Please refer to supplementary material for more details.

Data Partitions for Long-Tail Evaluation. The evaluation protocol of [46] is adopted by splitting the classes into many-shot, medium-shot, and few-shot. The splitting rule of [46] is used on iNaturalist. On CIFAR100-LT and AWA2-LT, the top and bottom 1/3 populated classes belong to many-shot and few-shot respectively, and the remaining to medium-shot.

Backbone Architectures. CIFAR100-LT and iNaturalist use the setup of [14], where ResNet32 [32] and ImageNet pre-trained ResNet50 are used respectively. For ImageNet-LT, ResNet10 is chosen as in [46]. For AWA2-LT, we use ResNet18.

Competence Level. Unless otherwise noted, the value of γ is cross-validated, i.e. the value of best performance on the validation set is applied to the test set.

Table 1. Ablations on iNaturalist-sub.

Method	leaf acc.	depth	hier. acc.	CPB	inference
Flat classifier	.163	1	.163	.163	-
RHC	.163	.58	.754	.537	BU
PI+STS	.174	.46	.913	.601	TD
PI+NCL	.185	.48	.904	.563	TD
PI+STS+NCL (Deep-RTC)	.181	.50	.899	.619	TD

Table 2. Comparisons to hierarchical classifiers.

Method	CIFAR100-LT	AWA2-LT	ImageNet-LT	inference
CNN-RNN [28]	.379	.882	.514	TD
B-CNN [64]	.366	.805	.511	TD/BU
HND [43]	.374	-	-	TD
NofE [2]	.373	.770	.463	BU
Deep-RTC	**.397**	**.894**	**.529**	TD

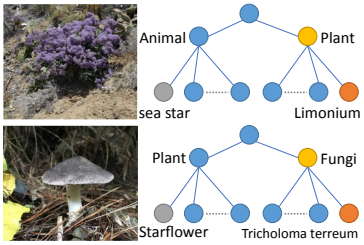

Fig. 4. Prediction of Deep-RTC (yellow) and flat classifier (gray) on two iNaturalist-sub images (orange: ground truth) (Color figure online).

5.2 Ablations

We started by evaluating how the different components of Deep-RTC - parameter inheritance (PI) regularization of (5), node-consistency loss (NCL) of (11), and stochastic tree sampling (STS) of (9) - affect the performance. Two baselines were used in this experiment. The first is a flat classifier, implemented with the standard softmax architecture, and trained to optimize classification accuracy. This is a representative baseline for the architectures used in the long-tailed recognition literature. The second is a hierarchical classifier derived from this flat classifier, by recursively adding class probabilities as dictated by the class taxonomy. This is denoted as the *recursive hierarchical classifier* (RHC). We refer to this computation of probabilities as *bottom-up* (BU) inference. This is opposite to the *top-down* (TD) inference used by most hierarchical approaches, where probabilities are sequentially computed from the root (top) to leaves (bottom) of the tree. The performance of the different classifiers was measured in multiple ways. CPB is the metric of (13). Leaf acc. is the classification accuracy at the leaves of the taxonomy. For a flat classifier, this is the standard performance measure. For a hierarchical classifier, it is the accuracy when intermediate rejections are not allowed. Hier acc. is the accuracy of a classifier that supports rejections, measured at the point where the decision is taken. In the example of Fig. 1 a decision of "dog" is considered accurate under this metric. Finally, depth is the average depth at which images are rejected, normalized by tree depth (e.g. 1 when no intermediate rejections are allowed).

CPB Performance: Table 1 shows that the flat classifier has very poor CPB performance because prediction at leaves requires the classifier to make decisions on tail classes where it is poorly trained. The result is a very large number of errors, i.e. images for which no label information is preserved. RHC, its bottom-up hierarchical extension, is a much better solution to long-tailed recognition. While most images are not classified at the leaves, both hierarchical accuracy and CPB increases dramatically. Nevertheless, RHC has weaker CPB performance than the combination of the PI architecture of Fig. 2 with either STS or NCL.

Table 3. Results on iNaturalist. Classes are discussed with popularity classes (many, medium and few- shot).

Method	metric	Many	Medium	Few	All
Softmax	CPB	0.76	0.67	0.62	0.66
CBLoss [14]		0.61	0.62	0.61	0.61
LDAM-SGD [7]		-	-	-	0.65
LDAM-DRW [7]		-	-	-	0.68
NCM [38]		0.61	0.64	0.63	0.63
cRT [38]		0.73	0.69	0.66	0.68
τ-norm [38]		0.71	0.69	0.69	0.69
Deep-RTC	CPB	**0.84**	**0.79**	**0.75**	**0.78**
	hier. acc.	0.92	0.91	0.89	0.90
	leaf freq.	0.71	0.56	0.48	0.54
	leaf acc.	0.76	0.67	0.60	0.64

Table 4. Results on ImageNet-LT.

Method	CPB
FSLwF [26]	0.28
Focal Loss [44]	0.31
Range Loss [62]	0.31
Lifted Loss [54]	0.31
OLTR [46]	0.36
Softmax	0.35
NCM [38]	0.36
cRT [38]	0.42
τ-norm [38]	0.41
Deep-RTC	**0.53**

Table 5. Comparisons to learning with rejection under different rejection rates (CPB).

		CIFAR100-LT				AWA2-LT			
Rej. Rate	Method	Many	Medium	Few	All	Many	Medium	Few	All
5%	RP [57]	**.779**	**.722**	.306	.404	**.977**	.963	.887	.914
	Deep-RTC	.773	.719	**.335**	**.416**	.975	**.978**	**.907**	**.931**
10%	RP [57]	**.793**	**.734**	.315	.416	**.980**	.966	.900	.924
	Deep-RTC	.789	.7314	**.344**	**.439**	.975	**.984**	**.929**	**.947**
20%	RP [57]	.816	.751	.328	.433	**.985**	.970	.916	.939
	Deep-RTC	**.833**	**.770**	**.393**	**.491**	.969	**.975**	**.943**	**.954**

Among these, the global regularization of STS is more effective than the local regularization of L_n. However, by combining two regularizations, they lead to the classifier (Deep-RTC) that preserves most information about the class label.

Performance Measures: The long-tailed recognition literature has focused on maximizing the accuracy of flat classifiers. Table 1 shows some limitations of this approach. First, all classifiers have very poor performance under this metric, with leaf acc. between 16% and 18%. Furthermore, as shown in Fig. 4, the labels can be totally uninformative of the object class. In the example, the flat classifier assigns the label of "sea star" ("star flower") to the image of the plant (mushroom) shown on the top (at the bottom). We are aware of *no* application that would find such labels useful. Second, all classifiers perform dramatically better in terms of hier. acc. For the practitioner, this means that they are accurate classifiers. Not expert enough to always carry the decision to the bottom of the tree, but reliable in their decisions. In the same example, Deep-RTC instead correctly assigns the images to the broader classes of "Plant" (top) and "Fungi" (bottom). Furthermore, Deep-RTC classifies 90% of the images correctly at this level! This could make it useful for many applications. For example, it could be used to automatically route the images to experts in these particular classes, for further labeling. Third, among TD classifiers, Deep-RTC pushes decisions furthest down the tree (e.g. 4% deeper than PI+STS). This makes it a better *expert* on iNaturalist-sub than its two variants, a fact captured by the proposed

CPB measure. Given all this, we believe that CPB optimality is much more meaningful than leaf acc. as a performance measure for long-tailed recognition.

5.3 Comparisons to Hierarchical Classifiers

We next performed a comparison to prior works in hierarchical classification with CPB in Table 2. These experiments show that prior methods have similar performance, without discernible advantage for TD or BU inference; however, they all underperform Deep-RTC. This is particularly interesting because these methods use networks more complex than Deep-RTC, adding branches (and parameters) to the backbone in order to regularize features according to the taxonomy. Deep-RTC simply implements a dynamic softmax classifier with the label encoding of Fig. 2. Instead, it leverages its dynamic ability and stochastic sampling to simultaneously optimize decisions for many tree cuts. The results suggest that this optimization over label sets is more important than shaping the network architecture according to the taxonomy. This is sensible since, under the Deep-RTC strategy, feature regularization is learned end-to-end, instead of hard-coded. Details of the compared methods are in the supplementary material.

5.4 Comparisons to Long-Tail Recognizers

A comparison to the state of the art methods from the long-tailed recognition is presented in Tables 3–4 for iNaturalist and ImageNet-LT respectively. More comparisons for other datasets are provided in the supplementary material. In all cases, Deep-RTC predicts *more bits* correctly (i.e. higher CPB), which beats the state of the art flat classifier by 9% on iNaturalist and 11% on ImageNet-LT. For iNaturalist, we also discuss other metrics by class popularity, where leaf freq. represents the frequency that samples are classified to leaves. A comparison to the standard softmax classifier shows that prior long-tailed methods improve performance CPB on few-shot classes but *degrade* for popular classes. Deep-RTC is the only method to consistently improve CPB performance for all levels of class popularity. It is also noted that, unlike the state of the art flat classifier, Deep-RTC does not have to sacrifice leaf acc. for the many-shot classes in order to accommodate few-shot classes where its performance will not be great anyway. Instead, it exits early for about half of the images of the few-shot classes and guarantees highly accurate answers for all classes (around 90% hier. acc.). This is similar to how humans treat the long-tail recognition problem.

5.5 Comparisons to Learning with Rejection

While the classifiers of the previous sections were allowed to reject examples at intermediate nodes, whenever feasible, they were not explicitly optimized for such rejection. Table 5 shows a comparison to a state-of-the-art flat realistic predictor (RP) [57], on CIFAR100-LT and AWA2-LT. In these comparisons, the percentage of rejected examples (rejection rate) is kept the same. The rejection

rate of Deep-RTC is the percent of examples rejected at the root node. Deep-RTC achieves the best performance for all rejection rates on both datasets, because it has the option of soft-rejecting, i.e. letting examples propagate until some intermediate tree node. This is not possible for the flat RP, which always faces an all or nothing decision. In terms of class popularity, Deep-RTC always has higher CPB for few-shot classes, and frequently considerable gains. For many and medium-shot classes, the two methods have the comparable performance on CIFAR100-LT. On AWA2-LT, RP has an advantage for many and Deep-RTC for medium-shot classes. This shows that the gains of Deep-RTC are mostly due to its ability to push images of low-shot classes as far down the tree as possible without forcing decisions for which the classifier is poorly trained.

6 Conclusion

In this work, a *realistic taxonomic classifier* (RTC) is proposed to address the long-tail recognition problem. Instead of seeking the finest-grained classification for each sample, we propose to classify each sample up to the level that the classifier is competent. Deep-RTC architecture is then introduced for implementing RTC with deep CNN and is able to 1) share knowledge between head and tail classes 2) align data hierarchy with model design in order to predict at all levels in the taxonomy, and 3) guarantee high prediction performance by opting to provide coarser predictions when samples are too hard. Extensive experiments validate the effectiveness of the proposed method on 4 long-tailed datasets using the proposed tree metric. This indicates that RTC is well suited for solving long-tail problem. We believe this opens up a new direction for long-tailed literature.

Acknowledgments. This work was partially funded by NSF awards IIS-1637941, IIS-1924937, and NVIDIA GPU donations.

References

1. iNaturalist 2018 Competition. https://github.com/visipedia/inat_comp
2. Ahmed, K., Baig, M.H., Torresani, L.: Network of experts for large-scale image categorization. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 516–532. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_32
3. Akata, Z., Reed, S., Walter, D., Lee, H., Schiele, B.: Evaluation of output embeddings for fine-grained image classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
4. Anaki, D., Bentin, S.: Familiarity effects on categorization levels of faces and objects. Cognition **111**, 144–149 (2009)
5. Anderson, J.: The adaptive nature of human categorization. Psychol. Rev. **98**, 409–429 (1991)
6. Buda, M., Maki, A., Mazurowski, M.: A systematic study of the class imbalance problem in convolutional neural networks. Neural Netw. **106** (2017). https://doi.org/10.1016/j.neunet.2018.07.011

7. Cao, K., Wei, C., Gaidon, A., Arechiga, N., Ma, T.: Learning imbalanced datasets with label-distribution-aware margin loss. In: Advances in Neural Information Processing Systems (NIPS) (2019)
8. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: Synthetic minority over-sampling technique. J. Artif. Int. Res. **16**(1), 321–357 (2002). http://dl.acm.org/citation.cfm?id=1622407.1622416
9. Chow, C.K.: An optimum character recognition system using decision functions. IRE Trans. Electron. Comput. **EC-6**, 247–254 (1957)
10. Chow, C.K.: On optimum recognition error and reject tradeoff. IEEE Trans. Inf. Theory **16**, 41–46 (1970)
11. Corbiére, C., Thome, N., Bar-Hen, A., Cord, M., Pérez, P.: Addressing failure detection by learning model confidence. In: Advances in Neural Information Processing Systems (NIPS) (2019)
12. Cortes, C., DeSalvo, G., Mohri, M.: Boosting with abstention. In: Advances in Neural Information Processing Systems (NIPS) (2016)
13. Cortes, C., DeSalvo, G., Mohri, M.: Learning with rejection. In: Ortner, R., Simon, H.U., Zilles, S. (eds.) ALT 2016. LNCS (LNAI), vol. 9925, pp. 67–82. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46379-7_5
14. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
15. Davis, J., Liang, T., Enouen, J., Ilin, R.: Hierarchical semantic labeling with adaptive confidence. In: International Symposium on Visual Computing (2019)
16. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
17. Deng, J., et al.: Large-scale object classification using label relation graphs. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 48–64. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_4
18. Deng, J., Krause, J., Berg, A.C., Fei-Fei, L.: Hedging your bets: optimizing accuracy-specificity trade-offs in large scale visual recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
19. Dong, Q., Gong, S., Zhu, X.: Class rectification hard mining for imbalanced deep learning. In: International Conference on Computer Vision (ICCV) (10 2017)
20. Drummond, C., Holte, R.: C4.5, class imbalance, and cost sensitivity: why under-sampling beats oversampling. Proceedings of the ICML 2003 Workshop on Learning from Imbalanced Datasets (2003)
21. El-Yaniv, R., Wiener, Y.: On the foundations of noise-free selective classification. J. Mach. Learn. Res. **11**, 1605–1641 (2010)
22. Fumera, G., Roli, F.: Support vector machines with embedded reject option. In: Lee, S.-W., Verri, A. (eds.) SVM 2002. LNCS, vol. 2388, pp. 68–82. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45665-1_6
23. Gal, Y., Ghahramani, Z.: Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: International Conference on Machine Learning (ICML) (2016)
24. Geifman, Y., El-Yaniv, R.: Selective classification for deep neural networks. In: Advances in Neural Information Processing Systems (NIPS) (2017)
25. Geifman, Y., El-Yaniv, R.: SelectiveNet: a deep neural network with an integrated reject option. In: International Conference on Machine Learning (ICML) (2019)

26. Gidaris, S., Komodakis, N.: Dynamic few-shot visual learning without forgetting. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
27. Goo, W., Kim, J., Kim, G., Hwang, S.J.: Taxonomy-regularized semantic deep convolutional neural networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 86–101. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_6
28. Guo, Y., Liu, Y., Bakker, E.M., Guo, Y., Lew, M.S.: CNN-RNN: a large-scale hierarchical image classification framework. Multimedia Tools Appl. **77**, 10251–10271 (2018)
29. He, H., Bai, Y., Garcia, E.A., Li, S.: ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322–1328 (2008)
30. Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005). https://doi.org/10.1007/11538059_91
31. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009). https://doi.org/10.1109/TKDE.2008.239
32. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
33. Horn, G.V., et al.: The iNaturalist species classification and detection dataset. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
34. Huang, C., Li, Y., Loy, C.C., Tang, X.: Learning deep representation for imbalanced classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
35. Jiang, H., Kim, B., Guan, M., Gupta, M.: To trust or not to trust a classifier. In: Advances in Neural Information Processing Systems (NIPS), pp. 5541–5552 (2018)
36. Johnson, K.: Impact of varying levels of expertise on decisions of category typicality. Memory Cogn. **29**, 1036–1050 (2001)
37. Johnson, K., Mervis, C.: Effects of varying levels of expertise on the basic level of categorization. J. Exp. Psychol. Gen. **126**(3), 248–77 (1997)
38. Kang, B., et al.: Decoupling representation and classifier for long-tailed recognition. In: International Conference on Learning Representations (ICLR) (2020)
39. Kim, H.J., Frahm, J.-M.: Hierarchy of alternating specialists for scene recognition. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11215, pp. 471–488. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01252-6_28
40. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, Citeseer (2009)
41. Krizhevsky, A., Hinton, G.: Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. IEEE Trans. Pattern Anal. Mach. Intell. **41**, 2251–2265 (2019)
42. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS) (2012)
43. Lee, K., Lee, K., Min, K., Zhang, Y., Shin, J., Lee, H.: Hierarchical novelty detection for visual object recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
44. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. IEEE Trans. Pattern Anal. Mach. Intell. **1**, 1 (2018)

45. Liu, Y., Dou, Y., Jin, R., Qiao, P.: Visual tree convolutional neural network in image classification. In: International Conference on Pattern Recognition (ICPR) (2018)
46. Liu, Z., Miao, Z., Zhan, X., Wang, J., Gong, B., Yu, S.X.: Large-scale long-tailed recognition in an open world. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
47. Mahajan, D., et al.: Exploring the limits of weakly supervised pretraining. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11206, pp. 185–201. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01216-8_12
48. Marszałek, M., Schmid, C.: Semantic hierarchies for visual object recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2007)
49. Miller, G.A.: WordNet: a lexical database for English. Commun. ACM **38**, 39–41 (1995)
50. Morgado, P., Vasconcelos, N.: Semantically consistent regularization for zero-shot recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
51. Salakhutdinov, R., Torralba, A., Tenenbaum, J.: Learning to share visual appearance for multiclass object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2011)
52. Shahbaba, B., Neal, R.M.: Improving classification when a class hierarchy is available using a hierarchy-based prior. Bayesian Anal. **2**(1), 221–238 (2007)
53. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
54. Song, H.O., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: IEEE Computer Vision and Pattern Recognition (CVPR) (2016)
55. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(56), 1929–1958 (2014). http://jmlr.org/papers/v15/srivastava14a.html
56. Tanaka, J., Taylor, M.: Object categories and expertise: is the basic level in the eye of the beholder. Cogn. Psychol. (1991). https://doi.org/10.1016/0010-0285(91)90016-H
57. Wang, P., Vasconcelos, N.: Towards realistic predictors. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11217, pp. 37–53. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01261-8_3
58. Wang, Y.X., Hebert, M.: Learning from small sample sets by combining unsupervised meta-training with CNNs. In: Advances in Neural Information Processing Systems (NIPS) (2016)
59. Wang, Y.-X., Hebert, M.: Learning to learn: model regression networks for easy small sample learning. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9910, pp. 616–634. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_37
60. Wang, Y.X., Ramanan, D., Hebert, M.: Learning to model the tail. In: Advances in Neural Information Processing Systems (NIPS) (2017)
61. Yan, Z., et al.: HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. In: International Conference on Computer Vision (ICCV) (2015)
62. Zhang, X., Fang, Z., Wen, Y., Li, Z., Qiao, Y.: Range loss for deep face recognition with long-tailed training data. In: International Conference on Computer Vision (ICCV) (2017)

63. Zhao, B., Fei-Fei, L., Xing, E.P.: Large-scale category structure aware image categorization. In: Advances in Neural Information Processing Systems (NIPS) (2011)
64. Zhu, X., Bain, M.: B-CNN: branch convolutional neural network for hierarchical classification. CoRR abs/1709.09890 (2017)
65. Zou, Y., Yu, Z., Vijaya Kumar, B.V.K., Wang, J.: Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11207, pp. 297–313. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01219-9_18