Supplementary Material: Solving Long-tailed Recognition with Deep Realistic Taxonomic Classifier

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

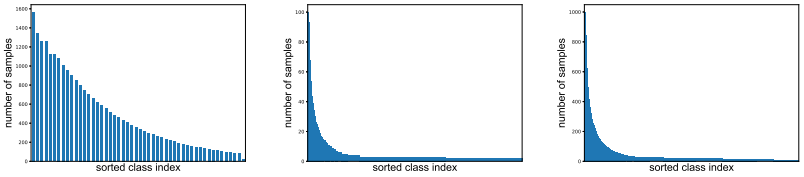
University of California, San Diego
{tzw001, pmaravil, pew062, chh279, nvasconcelos}@ucsd.edu

A Dataset details

While CIFAR100-LT, ImageNet-LT and iNaturalist (2018) are acquired from referenced papers [1,14,33,46], we curated AWA2-LT and iNaturalist-sub. AWA2-LT contains 25,622 training images and 3,000 test images. The training set is curated for imbalanced factor 0.01 (see Figure 1 (a)) and the test set is balanced. iNaturalist-sub is sampled from each class of the full iNaturalist dataset, with similar distribution, as shown in Figure 1 (b)-(c). Note that each class contains at least one image after subsampling.

B Details of baseline hierarchical classifiers

Four different hierarchical classifiers are evaluated and compared using the same backbone as Deep-RTC. **CNN-RNN [28]** uses the CNN as the feature extractor and an LSTM as the classifier to sequentially predict the label at each level of the hierarchy. **B-CNN [64]** adds convolutional branches at intermediate CNN layers for coarser level predictions, equating the output of the last layer to the leaf level predictions. During training, the loss is initially heavier for the coarse level predictions and then the weight gradually shifts to the leaf level loss. The top-down inference is adopted for the two-level CIFAR100-LT dataset. **NofE [2]** uses coarse level labels in a first stage and fine-grained level labels in a second



(a) AWA2-LT (b) iNaturalist-sub (c) iNaturalist

Fig. 1: Data distribution of 3 different datasets. X-axis is the sorted class index and y-axis is the number of training samples in each class. iNaturalist-sub remains similar distribution as iNaturalist.

stage. The second stage model is pre-trained with the first stage model and adds convolutional branches for each of the coarse labels. Since the original B-CNN and NofE networks are not designed for multi-level hierarchies, such as those of AWA2-LT and ImageNet-LT, we adopt labels of level 2 and 4 as coarse labels for these two datasets respectively. The bottom-up inference is adopted at test time for these two datasets. **HND [43]** uses a KL-Divergence to enforce a uniform distribution over the predictions of nodes outside of the ground truth path. Top-down inference is adopted in this experiment. For all methods, the competence level γ is selected on the validation set and applied to the test set.

C Implementation details

CIFAR100-LT We follow the setting of [14], training the proposed model for 200 epochs with the initial learning rate of 0.1 and batch size 128 with learning rate warm-up in the first 5 epochs. We used stochastic gradient descent with momentum 0.9 and weight decay 0.0005. The learning rate is decayed by a factor of 10 at epochs 160 and 180.
iNaturalist We train the model with batch size 128 with the initial learning rate 0.01. Following [38], we train for 200 epochs with the learning rate decayed by a factor of 10 after 100 epochs. Stochastic gradient descent with momentum 0.9 and weight decay 0.0001 is used to optimize this network.
AWA2-LT and ImageNet-LT We train for 120 and 90 epochs with initial learning rates 0.01 and 0.1 respectively, which are decayed by a factor of 10 every 40 and 30 epoch respectively.

We keep a small portion of the training data as the validation set to select hyperparameters, which is uniformly distributed similar to testing data. λ is set to 0.5 for iNaturalist and AWA2-LT, and λ is 1 for the other two datasets. Stochastic tree sampling is implemented with a dropout layer with the dropout rate of $p = 0.2$ for iNaturalist and CIFAR100-LT, and $p = 0.1$ for ImageNet-LT and AWA2-LT.

D Comparison to long-tail recognizers on CIFAR100-LT and AWA2-LT

Table 1 presents additional CPB results on CIFAR100-LT and AWA2-LT. Comparing to other representative long-tail recognizers, Deep-RTC achieves the highest CPB on both datasets. We also highlight that the performance gain of Deep-RTC increases with the dataset difficulty and the classification ability at each level. When the classification is easy (e.g. AWA2-LT), it can be done mostly at the leaves (leaf acc. is 89.4%), so there is little benefit exiting earlier in the tree. However, when the dataset is complex, the gains are significant. For example, the leaf acc. and hier. acc. for ImageNet-LT are 29.8% and 84.5% respectively. Hence, Deep-RTC only classifies 9.6% of samples to leaf level, but provides coarser predictions for most samples, which results in 11% gain in CPB. For CIFAR100-LT,

Table 1: Comparison to long-tail recognizers on CIFAR100-LT and AWA2-LT (CPB).

Method	CIFAR100-LT	AWA2-LT
Softmax	.381	.889
CBLoss [14]	.382	.890
Focal Loss [44]	.380	.885
LDAM [7]	.382	.880
Deep-RTC	**.397**	**.894**

Table 2: Convergence analysis.

percent of $l_0 - l_\infty$	0.2	0.4	0.7	0.9	0.999	1
CIFAR100-LT	6/6	12/12	42/40	161/161	187/189	200/200
AWA2-LT	2/2	3/3	7/8	19/26	99/103	120/120
ImageNet-LT	4/3	11/6	34/32	63/61	88/88	90/90

leaf acc. (39.5%) and hier. acc. (49.9%) are both low, so the information can be recovered correctly by exiting earlier is limited.

E Convergence analysis of stochastic tree sampling (STS)

Stochastic tree sampling (STS) has multiple roles in this paper. On one hand, STS regularizes the embedding, encouraging the embedding of an example to be close to the weight vectors of all its parent nodes. On the other, STS calibrates the probability of all possible classification heads, enabling more precise probability estimates at inference time, where different classification heads can consist of nodes from different tree heights.

We further discuss the convergence speed of the model with and without STS with the same number of training iterations. Table 2 presents the number of epochs needed for convergence. For each method, we compute the loss decrement between the initial and final losses (i.e. $d = l_0 - l_\infty$), and list the number of epochs needed to achieve certain percentages of decrement. For example, a percentage of 0.2 means a loss value of $l_0 - d * 0.2$; at the end of the training, the ratio is 1. In each column, the first value is the number of epochs needed for the model without STS, and the second one is for the model with STS. This comparison shows that the proposed method does not converge slower in practice.