LONG-TAILED RECOGNITION BY ROUTING DIVERSE DISTRIBUTION-AWARE EXPERTS

Xudong Wang[1,2], **Long Lian**[1,2], **Zhongqi Miao**[1,2], **Ziwei Liu**[3], **Stella X. Yu**[1,2]
[1]University of California, Berkeley, [2]International Computer Science Institute,
[3]Nanyang Technological University
xdwang@eecs.berkeley.edu, ziwei.liu@ntu.edu.sg,
{xdwang,longlian,zhongqi.miao,stellayu}@berkeley.edu

ABSTRACT

Natural data are often long-tail distributed over semantic classes. Existing recognition methods tend to focus on gaining performance on tail classes, often at the expense of losing performance on head classes and with increased classifier variance. The low tail performance manifests itself in large inter-class confusion and high classifier variance. We aim to reduce both the bias and the variance of a long-tailed classifier by RoutIng Diverse Experts (RIDE), consisting of three components: 1) a shared architecture for multiple classifiers (experts); 2) a distribution-aware diversity loss that encourages more diverse decisions for classes with fewer training instances; and 3) an expert routing module that dynamically assigns more ambiguous instances to additional experts. With on-par computational complexity, RIDE significantly outperforms the state-of-the-art methods by 5% to 7% on all the benchmarks including CIFAR100-LT, ImageNet-LT and iNaturalist 2018. RIDE is also a universal framework that can be applied to different backbone networks and integrated into various long-tailed algorithms and training mechanisms for consistent performance gains. Our code is publicly available at: https://github.com/frank-xwang/RIDE-LongTailRecognition.

1 INTRODUCTION

Real-world recognition often has a long-tail distribution: A few classes contain many instances, whereas most classes contain only a few instances. For extreme imbalanced classification, learning to discriminate among them is challenging, as the classifier training objective on the few tail instances can be easily overwhelmed by that for many head instances, whereas for the multitude of tail-class few-shot recognition tasks.

Long-tailed recognition is usually handled either by class re-balancing/re-weighting strategies which give more importance to tail instances (Cao et al., 2019; Kang et al., 2020; Liu et al., 2019) or by grouping methods, where long-tailed data are separated into groups by their class frequencies and models focusing on each individual group are combined to form a multi-expert framework (Zhou et al., 2020; Xiang et al., 2020). However, all these methods generally gain on tail classes at the cost of performance loss on head classes.

Here, we quantitatively analyze the long-tail classifier's performance change in terms of bias and variance analysis with respect to fluctuations in the training set: We randomly sample CIFAR100 (Krizhevsky, 2009) according to a long-tailed distribution a few times, train a model each time, and then estimate the per-class bias and variance of the classifiers.

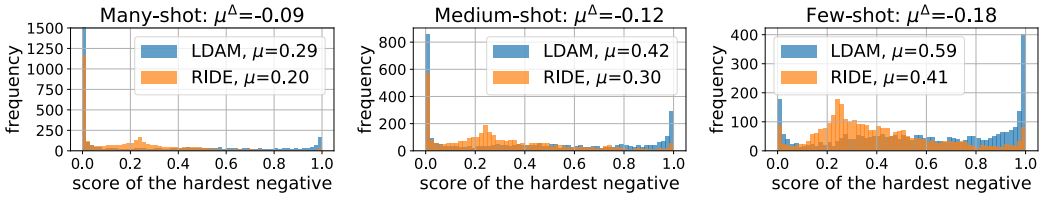
The prediction error of model h on instance x with output Y varies with the training data D. The expected variance with respect to D has a well-known bias-variance decomposition:

$$\text{Error}(x; h) = E[(h(x; D) - Y)^2] = \text{Bias}(h)^2 + \text{Variance}(h) + \text{irreducible error}. \tag{1}$$

The model bias measures the *accuracy* of the prediction with respect to the true value, the variance of the method measures the *stability* of the prediction, and the irreducible error measures the *precision* of the prediction and is irrelevant to the model h. This same concept can be extended to classification

	All			Many-shot			Med-shot			Few-shot		
	acc	bias	var	acc	bias	var	acc	bias	var	acc	bias	var
CE	31.6	0.60	0.47	57.3	0.28	0.35	28.2	0.61	0.51	6.3	0.94	0.57
τ-norm	35.8	0.52	0.49	55.9	0.28	0.37	33.2	0.53	0.52	16.1	0.78	0.60
cRT	36.4	**0.50**	0.50	51.3	0.32	0.41	38.6	**0.44**	0.50	17.0	0.76	0.61
LDAM	34.4	0.53	0.51	55.1	0.28	0.38	31.9	0.53	0.54	13.9	0.81	0.63
RIDE + LDAM	**40.5**	**0.50**	**0.42**	**60.5**	0.28	**0.30**	**38.7**	0.50	**0.44**	**20.1**	**0.74**	**0.52**

(a) Comparisons of the mean accuracy, per-class bias and variance of baseline methods and our RIDE method. Better (worse) metrics than the distribution-unaware cross entropy (CE) reference are marked in green (red).



(b) Histograms of the largest softmax probability of the other classes (the hardest negative) per instance.

Figure 1: The proposed method RIDE gains accuracies over SOTA by reducing both model bias and variance. **a)** These metrics are evaluated over 20 independently trained models, each on a random sampled set of CIFAR100 with an imbalance ratio of 100 and 300 samples for class 0. Compared to the standard cross-entropy (CE) classifier, existing SOTA methods almost always increase the variance and some reduce the tail bias at the cost of increasing the head bias. **b)** Compared with the many-shot classes, LDAM is more likely to confuse the tail classes with the hardest negative class, with an average score of 0.59. RIDE with LDAM can greatly reduce the confusion with the nearest negative class of each instance, especially for samples from the few-shot categories.

using 0-1 loss between the predicted class and the ground-truth label (Domingos, 2000). Below is a quick summary.

Let D be a set of training sets, x be a sample in the dataset. A loss function $\mathcal{L}(t; y)$ measures the cost of predicting $y \in Y$ when the true value is t. Assume the optimal prediction $y*$ for x is the prediction that minimizes $E_t[\mathcal{L}(t; y*)]$, where the subscript t denotes that the expectation is taken with respect to all possible values of t, weighted by their probabilities given x. The main prediction y_m for a loss function $\mathcal{L}$ over the training set D (which is treated as a random variable) is defined as $y_m = \text{argmin}_{y'} E_D[\mathcal{L}(y, y')]$. That is, the main prediction is the one that "differs least" from all the predictions in Y according to $\mathcal{L}$.

The bias is the loss incurred by the main prediction y_m relative to the optimal prediction $y*$, and the variance is the average loss incurred by predictions y relative to the main prediction y_m. We then evaluate bias and variance with zero one loss according to the evaluation metric in (Domingos, 2000) for classification, i.e. bias $= \mathcal{L}_{\text{zero-one}}(y*, y_m)$, variance $= E_D[\mathcal{L}_{\text{zero-one}}(y_m, y)]$.

Fig. 1a uses the standard cross-entropy (CE) classifier as a performance reference and compares three well-known long-tail methods, cRT and τ-norm (Kang et al., 2020) and LDAM (Cao et al., 2019), with our proposed method. The former adopts a two-stage optimization, first representation learning and then classification learning, whereas the latter is trained end-to-end with a marginal loss. On the basis of the preliminary experiments, following observations can be obtained:

1. On the mean accuracy: All the long-tail methods increase the overall, medium-shot, and few-shot accuracies, but these previous methods all decrease the many-shot accuracy. Our method increases accuracies on all splits.
2. On the model bias: All the long-tail methods reduce the overall, medium-shot, and few-shot bias. The reduction tends to be greater for the tail classes. Our method decreases bias more than other methods on the tail classes.
3. On the model variance: All the current long-tail methods increase the overall, many-shot, medium-shot, and few-shot variance, except cRT has a slight reduction for medium-shot. Our method reduces variances throughout the class spectrum.

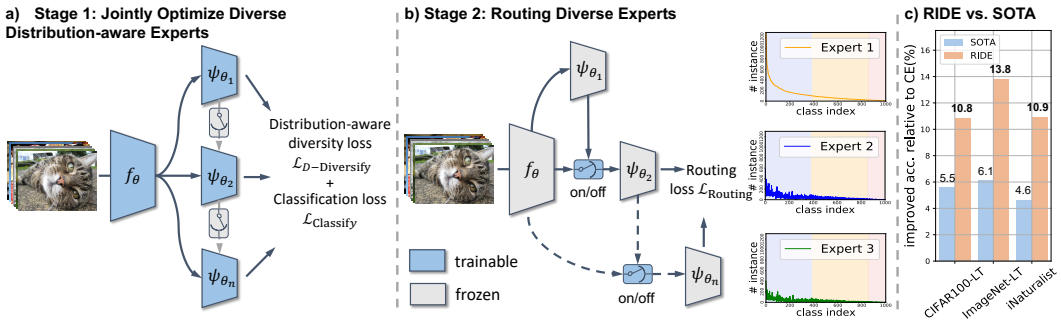


Figure 2: RIDE applies a two-stage optimization process. **a)** We first jointly optimize multiple diverse experts with distribution-aware diversity loss. **b)** An expert assignment module that could dynamically assign "ambiguous" samples to extra experts is trained in stage two. At test time, we combine the predictions of assigned experts to form a robust prediction. Since tail classes are inclined to be confused with other classes, by adding the expert assignment module, the data imbalance ratio for later experts can be automatically reduced without any distribution-aware loss, which allows focusing less on confident head classes and more on tail classes. **c)** RIDE outperforms SOTA methods (i.e. LFME (Xiang et al., 2020) for CIFAR100-LT, LWS (Kang et al., 2020) for ImageNet-LT and BBN (Zhou et al., 2020) for iNaturalist) on all experimented benchmarks.

That is, current long-tail methods sacrifice the head performance to improve the tail performance, and their models also become more unstable with respect to the training data variation. Our method is the only one that improves accuracies and reduces bias/variance throughout the class spectrum.

Fig. 1b provides further insight into the model bias by examining the largest softmax probability in the *other* classes: The smaller this value is, the less the confusion, and the lower the bias. It shows that the poor tail performance manifests itself at increasingly larger confusion and bias, and our method significantly reduces the confusion and bias for the tail classes.

We propose a novel multi-expert approach, called *RoutIng Diverse Experts* (RIDE), not only to reduce the model variance for all the classes, but also to reduce the model bias for the tail classes, both of which existing long-tail methods all fail to accomplish.

RIDE adopts multiple experts for model variance reduction and employs an additional distribution-aware diversity loss $\mathcal{L}_{\text{D-Diversity}}$ for reducing the model bias. With a dynamic expert routing module, RIDE assigns another trained and distinctive expert for a second (or third, ...) opinion when it is called for. Both the shared architecture for experts and the routing module effectively reduce the computational complexity of our multi-expert model to a level even lower than a baseline model with the same backbone.

RIDE delivers 5%$\sim$7% higher accuracies than the current SOTA methods on CIFAR100-LT, ImageNet-LT (Liu et al., 2019) and iNaturalist (Van Horn et al., 2018). RIDE is also a universal framework that can be applied to different backbone networks for improving existing long-tail algorithms such as focal loss (Lin et al., 2017), LDAM (Cao et al., 2019), τ-norm (Kang et al., 2020).

2 RELATED WORKS

Re-balancing/re-weighting. The classical way to achieve balance for long-tailed datasets is to control each class's sample frequencies based on class numbers such as the under- and over-sampling method (He & Garcia, 2009). Data augmentation is another common way to achieve sample balance. For example, Liu et al. (2020), Chu et al. (2020), and Kim et al. (2020) generate augmented samples in the feature space to supplement tail class samples. Besides sample-wise balancing, re-weighting the loss functions can also introduce learning balance by either putting larger weights on more challenging and sparse classes (Lin et al., 2017; Cui et al., 2019; Cao et al., 2019; Wu et al., 2020) or randomly ignoring gradients from head classes (Tan et al., 2020). However, both sample-wise and loss-wise balancing conceptually increase the learning focus on tail classes, which increases the sensitivity to small fluctuations of tail classes and greatly increases the variance as in Fig. 1.

Knowledge transfer. Another major direction to learning balance is transferring knowledge from head to tail classes (Liu et al., 2019; Zhu & Yang, 2020; Kang et al., 2020; Jamal et al., 2020; Wang et al., 2017). For example, the representative approaches in this direction, OLTR (Liu et al., 2019) and inflated memory (Zhu & Yang, 2020), harness memory banks to store and transfer mid- and high-level features from head classes to enhance tail classes' feature generalization. However, this line of work usually do not have effective controls over the transferring process, and therefore would sacrifice the performance of head classes.

Ensemble and grouping. A more related direction of balanced learning to this project is ensemble and grouping, in which samples are usually separated into different groups based on the position of the class from head to tail and individual models with focus on each group are ensembled at the end to form a multi-expert framework. BBN (Zhou et al., 2020) has a two-branch architecture with an adaptive fusion procedure at the end, where one branch focuses on the head classes (by directly learning from an imbalanced dataset), and the other focuses on the tail classes (by balancing the dataset with sampling techniques). LFME (Xiang et al., 2020) takes a step further by distilling a unified model from multiple teacher models; each focuses on classifying a relatively balanced group of the dataset (e.g., many-shot classes, medium-shot classes, and few-shot classes). BBN and LFME still suffer from performance loss in head classes because each expert do not have a balanced access to the whole dataset, which damages the overall generalizability, especially head classes. In our approach, we apply a diverse distribution-aware individual loss with instance-balanced sampling to each routed expert, so that the information between experts are shared to boost generalizability.

RIDE has several appealing properties over traditional ensemble methods: **1)** RIDE requires much fewer parameters and computational cost, RIDE's model complexity can even be smaller than the baseline model with higher accuracy. **2)** By adding the expert assignment module, not all experts are activated for all samples. The dynamic collaboration between experts enables RIDE to be more efficient. **3)** All these experts are jointly optimized, which reduces the cost for running back-propagation and optimization steps. **4)** We share partial convolutional layers of these experts that are considered relatively task-agnostic and reduce the channel dimensions of non-shared layers. Since the capability of each expert is lower, each expert is less easier to be over-fitted on the tail classes.

3 RIDE: Routing Diverse Distribution-Aware Experts

3.1 Overall framework

The multi-expert framework consists of two components as in Fig. 2a: f_θ, shared part across experts, and $\Psi = \{\psi_{\theta_1}, ..., \psi_{\theta_n}\}$, which is expert-independent and evaluated with the output of f_θ.

Although our method is applicable to multiple backbone networks, which is also indicated in our experiments, we take ResNet as an example to illustrate our design for its commonness. We exploit a property of convolutional neural networks in the design of the multi-expert framework: Since early CNN layers typically contain filters that generally process only task-agnostic information of an image, as indicated by the practice of freezing early layers in transfer learning, we share the first two stages of ResNet across all experts, i.e. f_θ. Later stages, including the fully connected layer, are independent between experts, i.e. ψ_{θ_i}, where $i \in [1, n]$. In addition, in order to further cut down the overall model complexity, the number of filters in ψ_{θ_i} is reduced by 1/4.

Then, to obtain logits of each experts, we first get the intermediate representation from f_θ, and then pass it into each ψ_{θ_i} independently to get logits. All these n experts are co-trained on long-tailed data with the proposed distribution aware diversity loss $\mathcal{L}_{\text{D-Diversify}}$ and a classification loss $\mathcal{L}_{\text{Classify}}$, such as cross-entropy and LDAM.

Expert assignment module is added to dynamically assign additional cascaded experts to hard samples. Since a weak expert is sufficient to provide a high-quality representation for easy samples, not all samples require extra knowledge from additional experts. The expert assignment module is trained with a separate stage on the top of frozen feature extractor as in Fig. 2b.

Making joint decisions with geometric mean. During testing time, assume m, where $m \leq n$, is the number of experts assigned to the tested sample, we handle the outputs of these models by taking arithmetic mean on logits of m experts, which produces logits with the same ranking as taking the geometric mean on the probabilities, followed by a softmax layer to calculate the probability of the

jth class: $p_j = \mathrm{softmax}(\frac{1}{m}\sum_{i=1}^{m}\psi_{\theta_i}(f_\theta(x)))$. In our setting, this approach performs better than taking the arithmetic mean on the probabilities, i.e. $p_j = \frac{1}{m}\sum_{i=1}^{m}\mathrm{softmax}(\psi_{\theta_i}(f_\theta(\vec{x})))$.

3.2 Diverse Distribution-Aware Individual Loss

Previous multi-branch methods such as BBN apply a classification loss (e.g. cross-entropy) on aggregated output of models. While this is intuitive due to same formula in training and evaluation, in our empirical experiments, we found that applying this method is not sufficient for the best performance. We hypothesize that taking the loss over the geometric mean in Section 3.1 leads to correlated experts, which prevents the experts to have different knowledge. In order to make the experts model decoupled, we present a loss function with two specifically-designed components.

Individual loss: decoupling the training of multiple experts. We define the loss function above as "collaborative loss", with expression: $\mathcal{L}^{\text{collaborative}} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\frac{1}{n}\sum_{j=1}^{n}\psi_{\theta_j}(f_\theta(\vec{x_i})), y_i)$, where N is the mini-batch size, $\mathcal{L}$ is the classification loss. This loss is shown to behave well in several recently proposed multi-expert models. However, in our multi-experts setting, this method only achieves accuracy comparable to a single-expert model with roughly the same number of parameters, which indicates that the collaborative loss does not train experts to have diverse knowledge to complement other experts in decision making, i.e. the collaborative loss trains correlated experts. This means the best performance of multi-experts overall is not shown.

To reduce the effect of correlated experts, we propose a new loss function, "individual loss", which applies the loss in the middle of the module as soon as each expert's logits are computed. Since the loss calculation process does not depend on post-aggregation output, it results in almost no inter-expert relationship expect the shared parameters. The formal expression of it, in training time, is: $\mathcal{L}^{\text{individual}} = \frac{1}{nN}\sum_{i=1}^{N}\sum_{j=1}^{n}\mathcal{L}_i(\psi_{\theta_j}(f_\theta(\vec{x_i})), y_i)$. The test time formulation is the same as the one mentioned in Section 3.1. Although it requires the model to have more than one output, which might add a little implementation difficulty, this loss function achieves a better accuracy and contributes a large amount of improvements in most of our experiments.

A diversified distribution-aware loss is proposed to penalize the inter-expert correlation further since we discover from experiments that although the individual loss, as well as the random initialization, already leads to branches that are diverse, adding more diversity with a external loss benefits an additional small amount most of the time. Since a complex model needs to be fitted on a few tail class samples, it will capture the noise along with the underlying pattern in data and thus we encourage diversity in tail classes to alleviate the influence of noise, we reference the idea of temperature in contrastive loss (Hadsell et al., 2006; Wu et al., 2018) to enable the diversity loss to be distribution aware, formulated as:

$$\mathcal{L}^{i}_{\text{D-Diversify}} = -\frac{\lambda}{n-1}\sum_{j\neq i}^{n} D_{\mathrm{KL}}(\phi^i(\vec{x},\vec{T}), \phi^j(\vec{x},\vec{T})) \tag{2}$$

where $\phi^i(\vec{x},\vec{T}) = \mathrm{softmax}(\psi_{\theta_i}(f_\theta(\vec{x}))/\vec{T})$, i is the expert index, $D_{\mathrm{KL}}(\cdot,\cdot)$ is the KL-Divergence ($D_{\mathrm{KL}}(P,Q) = \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)}$), λ is the balancing factor between diversity and classification loss between ground truth label $\vec{y}$ and model predictions, $\vec{T}$ is temperatures of each sample and division is performed as element-wise division, and $\mathcal{X}$ is the probability space of $P(x)$ and $Q(x)$.

The optimization objective of $\mathcal{L}^{i}_{\text{D-Diversify}}$ is to encourage the diversity between expert i and other experts. Choosing a small temperature would assign large negative gradients (i.e. smaller overall gradients) to samples with "divergent" predictions and very small negative gradients (i.e. larger overall gradients) to samples with "homogeneous" predictions. $\mathcal{L}^{i}_{\text{D-Diversify}}$ could also be used as a regularization term. The temperature T_i for samples in class i can be calculated by:

$$T_i = \eta\psi_i + \eta(1-\max(\Psi)); \Psi = \{\psi_1, ..., \psi_C\} = \{\gamma\cdot C\cdot\frac{n_i}{\sum_{k=1}^{C} n_k} + (1-\gamma)\}_{i=1}^{C} \tag{3}$$

where C is the number of classes, γ is the balancing factor, n_i is the number of samples of class i, and η is the base temperature. This way, we give lower temperature to tail classes, which generates higher probability for the tail classes in distributions that we apply KL Divergence on, encouraging

more diversity in tail classes. Similar to the practice of deferred reweight (Cao et al., 2019), we enable temperatures only after the network has been trained for several epochs to allow the feature extractor to learn stabilized features.

The total loss presented above can be formulated as:

$$\mathcal{L}_{\text{Total}} = \frac{1}{nN}\sum_{i=1}^{N}\sum_{j=1}^{n}\mathcal{L}^{j}_{\text{Classify}}(\phi^{j}(\vec{x}), y) - \frac{\lambda}{n-1}\sum_{j\neq k}^{n} D_{\text{KL}}(\phi^{j}(\vec{x}, \vec{T}), \phi^{k}(\vec{x}, \vec{T})) \tag{4}$$

where j and k are the expert indices, $\mathcal{L}^{j}_{\text{Classify}}(.,.)$ can be LDAM loss, focal loss, etc., depending on the training mechanisms we choose.

3.3 Routing Diversified Experts

An expert assignment module that dynamically routes cascaded experts is proposed to further reduce the overall computational complexity. The expert assignment decision is based on the information from the logits and the input feature. The input feature is normalized to stabilize training process and is transformed by a fully-connected layer to reduce dimension and aggregate information. We only take the top z scores of logits, since other logits are generally too small to provide much information about the "ambiguity" of the sample. z is commonly set to a number ranging from 10 to 50. The module has negligible computation cost and negligible additional parameters.

Let $\vec{l_i}$ and $\vec{v_i}$ be the top z logits and the normalized feature of expert i, respectively. The binary output of expert assignment module is: $y_{\text{ea}} = \mathbf{W}_2(\vec{l_i} \oplus \sigma(\mathbf{W}_1\vec{v_i}))$, where $\oplus$ denotes concatenation, $\sigma(\cdot)$ denotes ReLU function, $\mathbf{W}_1 \in \mathbb{R}^{d'\times d}$, $\mathbf{W}_2 \in \mathbb{R}^{1\times(d'+z)}$, and d is the dimension of input feature. d' is set to 16, as a too large d' may cause overfitting and unstable optimization.

The expert assignment module is optimized with the **routing loss**, a weighted variant of binary cross entropy loss:

$$\mathcal{L}_{\text{Routing}} = -\omega_{\text{p}} y \log(\frac{1}{1+e^{-y_{\text{ea}}}}) - \omega_{\text{n}}(1-y)\log(1-\frac{1}{1+e^{-y_{\text{ea}}}}) \tag{5}$$

where the ground truth y is constructed as: if the current expert does not predict the sample correctly but one of the next experts gives correct prediction, the ground truth is set to 1 (considered as a positive sample), otherwise it is 0. ω_{p} and ω_{n} are the re-weighting factor (set to 100 for all positive samples and 1 for all negative samples). The expert assignment module is trained on a separate stage, with all other parameters frozen. Each expert, except the last one, has an independent expert assignment module with shared $\mathbf{W}_2$. Therefore, the number of expert assignment modules is $n-1$.

Since we use a binary cross entropy loss with logits input in training, the output of this module is the logits that decide whether we need to activate more experts, and the predicted probability can be obtained by appending a sigmoid function at test time. If the predicted probability is greater or equal to 0.5, which means that our predictor gives larger probability to having additional experts, we decide to proceed to the next expert. In case that one needs to manually adjust the proportion of samples passing through each expert, i.e. the "easiness for each expert assignment module to be satisfied", one could adjust the positive weight of the binary cross entropy loss, ω_{p}, at training time. As observed in experiments, using a fixed positive weight ($\omega_{\text{p}} = 100$) across all our experiments leads to a good trade-off of our EA module in accuracy and computational complexity.

Self-distillation step is optional but recommended if further improvements (about 0.4%$\sim$0.8% for most experiments) are desired. Unlike previous methods with fixed number of experts (e.g. BBN, LFME), the total number of experts can be arbitrarily adjusted to balance accuracy and computation cost. This enables self-distillation from a model with more (6 in our setting) experts to the same model with fewer experts to obtain further improvements. We choose knowledge distillation (Hinton et al., 2015) by default. The implementation details and comparison on various distillation algorithms such as CRD (Tian et al., 2019) are investigated in Appendix Section A.2.

Table 1: Top-1 accuracy comparison with state-of-the-arts on **CIFAR100-LT** with an imbalance ratio of 100. Compared with BBN (Zhou et al., 2020) and LFME (Xiang et al., 2020), which also contain multiple experts (or branches), RIDE (2 experts) outperforms them by a large margin with fewer GFlops. The relative computation cost (averaged on testing set) with respect to the baseline model and absolute improvements against SOTA (colored in green) are reported. † denotes our reproduced results with released code. ‡ denotes results copied from (Cao et al., 2019).

Methods	MFlops	Acc. (%)	Many	Med	Few
Cross Entropy (CE) ‡	69.5 (1.0x)	38.3	-	-	-
Cross Entropy (CE) †	69.5 (1.0x)	39.1	66.1	37.3	10.6
Focal Loss ‡ (Lin et al., 2017)	69.5 (1.0x)	38.4	-	-	-
OLTR † (Liu et al., 2019)	-	41.2	61.8	41.4	17.6
LDAM + DRW (Cao et al., 2019)	69.5 (1.0x)	42.0	-	-	-
LDAM + DRW † (Cao et al., 2019)	69.5 (1.0x)	42.0	61.5	41.7	20.2
BBN (Zhou et al., 2020)	74.3 (1.1x)	42.6	-	-	-
τ-norm † (Kang et al., 2020)	69.5 (1.0x)	43.2	65.7	43.6	17.3
cRT † (Kang et al., 2020)	69.5 (1.0x)	43.3	64.0	44.8	18.1
M2m (Kim et al., 2020)	-	43.5	-	-	-
LFME (Xiang et al., 2020)	-	43.8	-	-	-
RIDE (2 experts)	**64.8 (0.9x)**	47.0 (+3.2)	67.9	48.4	21.8
RIDE (3 experts)	77.8 (1.1x)	48.0 (+4.2)	68.1	49.2	23.9
RIDE (4 experts)	91.9 (1.3x)	**49.1 (+5.3)**	**69.3**	**49.3**	**26.0**

4 EXPERIMENTS

4.1 DATASETS AND IMPLEMENTATIONS

We conduct experiments on three major long-tailed recognition benchmarks and different backbone networks to prove the effectiveness and universality of RIDE:

CIFAR100-LT (Cao et al., 2019): The long-tailed version of CIFAR100 follows an exponential decay in sample sizes across different classes. We conduct experiments on CIFAR100-LT with an imbalance factor of 100, with the ResNet-32 (He et al., 2016) as a backbone network.

ImageNet-LT (Liu et al., 2019): RIDE with ResNet-50 and ResNeXt-50 (Xie et al., 2017) are experimented on ImageNet-LT. More details and experiments on other backbones are listed in appendix.

iNaturalist (Van Horn et al., 2018): We use ResNet-50 as the backbone network and apply the same training recipe as ImageNet-LT except with batch size 512 as Kang et al. (2020) does.

4.2 EXPERIMENTAL RESULTS

CIFAR100-LT. Table 1 shows that RIDE outperforms state-of-the-art methods by a large margin on CIFAR100-LT. The average computational cost is even about 10% less than baseline models when we only apply 2 cascaded experts as in BBN. Compared with LFME (Xiang et al., 2020) and BBN (Zhou et al., 2020), which also apply multiple experts, RIDE significantly surpasses both by more than 5.3% and 6.5%, respectively. Since LDAM is end-to-end optimized, we choose it as the default training method for simplicity, unless otherwise noticed.

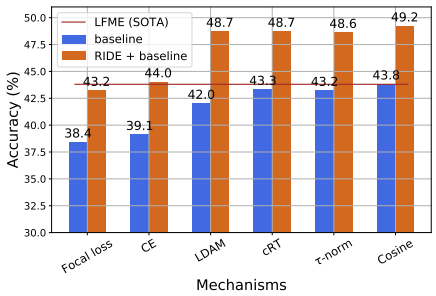


Figure 3: **Extend RIDE to various long-tailed recognition methods.** Consistent improvements can be observed on CIFAR100-LT, which illustrates that the proposed method can be applied to various training mechanisms, either methods that are end-to-end (e.g. LDAM) or require another stage of process (e.g. cRT and τ-norm). By using RIDE, cross-entropy loss (without any re-balancing strategies) can even outperforms current SOTA method LFME. Although higher accuracy can be obtained using distillation, we did not apply it here.

Table 2: Top-1 accuracy comparison with state-of-the-art methods on **ImageNet-LT** (Liu et al., 2019) with ResNet-50 and ResNeXt-50. RIDE achieves consistent performance improvements on various backbones. Results marked with † are copied from (Kang et al., 2020). We compare GFlops against the baseline model. Detailed results on each split are listed in appendix materials.

Methods	ResNet-50		ResNeXt-50	
	GFlops	Acc. (%)	GFlops	Acc. (%)
Cross Entropy (CE) †	4.11 (1.0x)	41.6	4.26 (1.0x)	44.4
OLTR † (Liu et al., 2019)	-	-	-	46.3
NCM (Kang et al., 2020)	4.11 (1.0x)	44.3	4.26 (1.0x)	47.3
τ-norm (Kang et al., 2020)	4.11 (1.0x)	46.7	4.26 (1.0x)	49.4
cRT (Kang et al., 2020)	4.11 (1.0x)	47.3	4.26 (1.0x)	49.6
LWS (Kang et al., 2020)	4.11 (1.0x)	47.7	4.26 (1.0x)	49.9
RIDE (2 experts)	**3.71 (0.9x)**	54.4 (+6.7)	**3.92 (0.9x)**	55.9 (+6.0)
RIDE (3 experts)	4.36 (1.1x)	54.9 (+7.2)	4.69 (1.1x)	56.4 (+6.5)
RIDE (4 experts)	5.15 (1.3x)	**55.4** (+7.7)	5.19 (1.2x)	**56.8** (+6.9)

Integrating RIDE with various methods. Fig. 3 indicates that our method will likely be able to benefit from the future advancements to the loss function and training process. Consistent improvements can be observed. Cosine classifier is a baseline that we constructed in which we normalize the weights of the classifier and apply resample in the classifier retrain stage, similar to cRT. Since two-stage methods require an additional training stage, we use LDAM as our default choice.

ImageNet-LT. We further evaluate RIDE on ImageNet-LT with various backbones as in Table 2. Compared with current SOTA methods, LWS and cRT, RIDE achieves new state-of-the-art results and outperforms SOTA by more than 7.7% with ResNet-50. ResNeXt-50 is based on group convolution (Xie et al., 2017), which divides all filters into several groups and aggregates information from multiple groups. ResNeXt-50 generally performs better than ResNet-50 on multiple tasks. Using ResNeXt-50, we can see a performance improvement of 6.9%.

Table 3: Comparison with state-of-the-art methods on **iNaturalist** (Van Horn et al., 2018). RIDE outperforms current SOTA BBN, which also contains multiple "experts", by a large margin on many-shot classes. Results marked with † are from BBN (Zhou et al., 2020) and Decouple (Kang et al., 2020). BBN's results are from the released checkpoint. Relative improvements to SOTA result of each split (colored with gray) are also listed, with the largest boost from few-shot classes.

Methods	GFlops	All	Many	Medium	Few
CE †	4.14 (1.0x)	61.7	72.2	63.0	57.2
CB-Focal †	4.14 (1.0x)	61.1	-	-	-
OLTR	4.14 (1.0x)	63.9	59.0	64.1	64.9
LDAM + DRW †	4.14 (1.0x)	64.6	-	-	-
cRT	4.14 (1.0x)	65.2	69.0	66.0	63.2
τ-norm	4.14 (1.0x)	65.6	65.6	65.3	65.9
LWS	4.14 (1.0x)	65.9	65.0	66.3	65.5
BBN	4.36 (1.1x)	66.3	49.4	70.8	65.3
RIDE (2 experts)	**3.67 (0.9x)**	71.4 (+5.1)	70.2 (+1.2)	71.3 (+0.5)	71.7 (+5.8)
RIDE (3 experts)	4.17 (1.0x)	72.2 (+5.9)	70.2 (+1.2)	72.2 (+1.4)	72.7 (+6.8)
RIDE (4 experts)	4.51 (1.1x)	**72.6** (+6.3)	**70.9** (+1.9)	**72.4** (+1.6)	**73.1** (+7.2)

iNaturalist. iNaturalist 2018 is a naturally imbalanced fine grained dataset with 8,142 categories. Table 3 shows that RIDE outperforms current SOTA by 6.3%. Surprisingly, RIDE obtains very similar results in many-shots, medium-shots and few-shots, which is a very ideal result for the long tailed recognition task. Current SOTA method BBN also uses multiple experts. Compared with BBN, which significantly decreases the performance on many-shot classes by about 23%, RIDE increases the accuracy on few-shot without sacrificing the performance on many-shot categories.

Contribution of each component of RIDE. RIDE is jointly trained with $\mathcal{L}_{\text{D-Diversify}}$ and $\mathcal{L}_{\text{Classify}}$, we use LDAM as $\mathcal{L}_{\text{Classify}}$ by default. Table 4 shows that the architectural change from the original ResNet-32 to the RIDE variant with $2 \sim 4$ experts contributes $2.7\% \sim 4.3\%$ accuracy improvements. The change from applying our classification loss naively to model output (collaborative loss) to applying to each expert as an individual loss brings 1.5% improvements. Adding diversity further

Table 4: **Ablation studies on the effectiveness of each component** on CIFAR100-LT. LDAM (Cao et al., 2019) is used as our classification loss. The first 3 RIDE models only have architectural change without changes in training method. The performance without $\mathcal{L}_{\text{Individual}}$ checked indicates directly applying classification loss onto the final model output, which is the mean expert logits. This is referred to as collaborative loss above. In contrast, if $\mathcal{L}_{\text{Individual}}$ if checked, we apply individual loss to each individual expert. The difference between collaborative loss and individual loss is described above. Knowledge distillation step is optional if further improvements are desired. Various knowledge distillation techniques are compared in the appendix.

Methods	#expert	$\mathcal{L}_{\text{Individual}}$	$\mathcal{L}_{\text{D-Diversify}}$	EA	distill	GFlops	Acc. (%)
LDAM + DRW	1						42.0
RIDE	2					1.1x	44.7 (+2.7)
	3					1.5x	46.1 (+4.1)
	4					1.8x	46.3 (+4.3)
	4	✓				1.8x	47.8 (+5.8)
	4	✓	✓			1.8x	48.7 (+6.7)
	4	✓	✓		✓	1.8x	**49.3** (+7.3)
	4	✓	✓	✓	✓	1.3x	49.1 (+7.1)

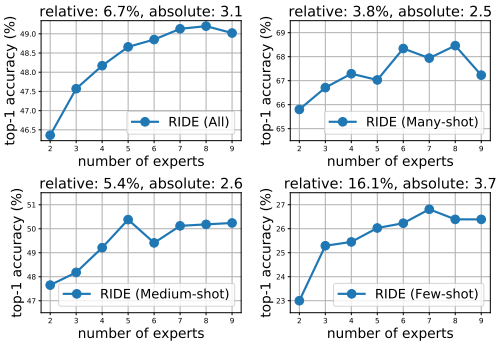


Figure 4: **# experts vs. top-1 accuracy for each split** (All, Many/Medium/Few) of CIFAR100-LT. Compared with the many-shot split, which is 3.8% relatively improved by adding more experts, the few-shot split can get more benefits, that is, a relative improvement of 16.1%.

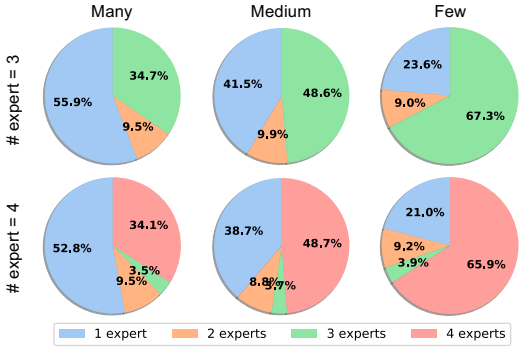


Figure 5: **The proportion of the number of experts** allocated to each split of CIFAR100-LT. For RIDE with 3 or 4 experts, more than half of many-shot instances only require one expert. On the contrary, more than 76% samples of few-shot classes require opinions from additional experts.

improves about 0.9%. The computation cost is greatly reduced by adding the expert assignment module. Knowledge distillation from RIDE with 6 experts obtains another 0.6% increase. All these coherent components of RIDE enable it to get 7.1% increase compared with baseline LDAM.

Influence of expert number. Fig. 4 shows the influence of expert number for each split. Compared with many-shot, which only obtains 3.8% relative improvements by using 8 experts, the accuracy of few-shot classes is relative increased by more than 16%, which indicates that few-shot classes can enjoy more benefits from using more experts. No distillation is applied in this comparison.

Number of experts allocated to each split. Fig. 5 shows that: Most instances in the many-shot classes are assigned only 1 expert, whereas those in few-shot classes are often assigned more experts. The low confidence in low-shot instances requires the model to seek a second (or a third, ...) opinion.

5 Summary

We propose a novel multi-expert approach for long-tailed recognition. It trains partially shared diverse distribution-aware experts and routes an instance to additional experts when necessary, with computational complexity comparable to a single expert. RIDE significantly outperforms SOTA methods by a large margin on all the benchmarks including CIFAR100-LT, ImageNet-LT, and iNaturalist. It can also be applied to various backbones and methods with consistent performance gains.

REFERENCES

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, pp. 1567–1578, 2019.

Peng Chu, Xiao Bian, Shaopeng Liu, and Haibin Ling. Feature space augmentation for long-tailed data. *arXiv preprint arXiv:2008.03673*, 2020.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9268–9277, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Pedro Domingos. A unified bias-variance decomposition. In *Proceedings of 17th International Conference on Machine Learning*, pp. 231–238, 2000.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1735–1742, 2006.

Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7610–7619, 2020.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Learning imbalanced datasets with label-distribution-aware margin loss. In *International Conference on Learning Representations*, pp. 1567–1578, 2020.

Jaehyung Kim, Jongheon Jeong, and Jinwoo Shin. M2m: Imbalanced classification via major-to-minor translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13896–13905, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Jialun Liu, Yifan Sun, Chuchu Han, Zhaopeng Dou, and Wenhui Li. Deep representation learning on long-tailed data: A learnable embedding augmentation perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2970–2979, 2020.

Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2537–2546, 2019.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4004–4012, 2016.

Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision*, pp. 268–284, 2018.

Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11662–11671, 2020.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2019.

Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1365–1374, 2019.

Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The iNaturalist species classification and detection dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8769–8778, 2018.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems*, pp. 7029–7039, 2017.

Tong Wu, Qingqiu Huang, Ziwei Liu, Yu Wang, and Dahua Lin. Distribution-balanced loss for multi-label classification in long-tailed datasets. In *Proceedings of the European Conference on Computer Vision*, pp. 162–178. Springer, 2020.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

Liuyu Xiang, Guiguang Ding, and Jungong Han. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *Proceedings of the European Conference on Computer Vision*, pp. 247–263. Springer, 2020.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1492–1500, 2017.

Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5409–5418, 2017.

Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. BBN: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9719–9728, 2020.

Linchao Zhu and Yi Yang. Inflated episodic memory with region self-attention for long-tailed visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4344–4353, 2020.

A APPENDIX

A.1 DATASETS AND IMPLEMENTATIONS

We conduct experiments on three major long-tailed recognition benchmarks and different backbone networks to prove the effectiveness and universality of RIDE:

1.**CIFAR100-LT** (Krizhevsky, 2009; Cao et al., 2019): The original version of CIFAR-100 contains 50,000 images on training set and 10,000 images on validation set with 100 categories. The long-tailed version of CIFAR-100 follows an exponential decay in sample sizes across different categories. We conduct experiment on CIFAR100-LT with an imbalance factor of 100, i.e. the ratio between the most frequent class and the least frequent class.

To make fair comparison with previous works, we follow the training recipe of (Cao et al., 2019) on CIFAR100-LT. We train the ResNet-32 (He et al., 2016) backbone network by SGD optimizer with a momentum of 0.9. CIFAR100-LT is trained for 200 epochs with standard data augmentations (He et al., 2016) and a batch size of 128 on one RTX 2080Ti GPU. The learning rate is initialized as 0.1 and decayed by 0.01 at epoch 120 and 160 respectively.

2.**ImageNet-LT** (Deng et al., 2009; Liu et al., 2019): ImageNet-LT is constructed by sampling a subset of ImageNet-2012 following the Pareto distribution with the power value $\alpha = 6$ (Liu et al., 2019). ImageNet-LT consists of 115.8k images from 1,000 categories, with the largest and smallest categories containing 1,280 and 5 images, respectively.

Multiple backbone networks are experimented on ImageNet-LT, including ResNet-10, ResNet-50 and ResNeXt-50 (Xie et al., 2017). All backbone networks are trained with a batch size of 256 on 8 RTX 2080Ti GPUs for 100 epochs using SGD with an initial learning rate of 0.1 decayed by 0.1 at 60 epochs and 80 epochs. We utilize standard data augmentations as in (He et al., 2016).

3.**iNaturalist** (Van Horn et al., 2018): The iNaturalist-2018 dataset is an imbalanced datasets with 437,513 training images from 8,142 classes with a balanced test set of 24,426 images. We use ResNet-50 as the backbone network and apply the same training recipe as ImageNet-LT, except that we use a batch size of 512.

A.2 ADDITIONAL EXPERIMENTS

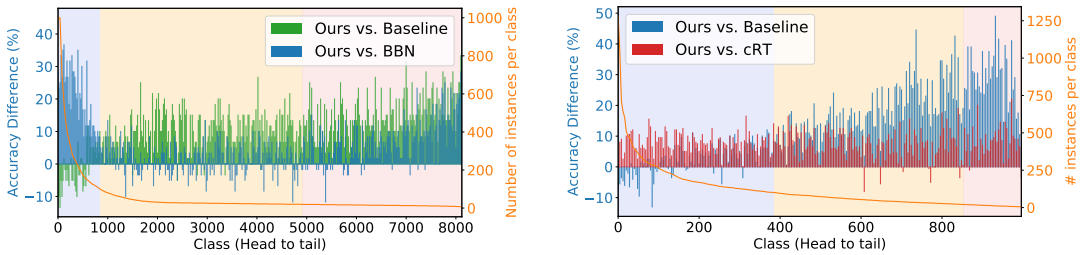


Figure 6: The absolute accuracy difference of RIDE (blue) over *iNaturalist*'s current state-of-the-art method BBN (Zhou et al., 2020) (**left**) and *ImageNet-LT*'s current state-of-the-art method cRT (Kang et al., 2020) (**right**). RIDE improves the performance of few- and medium-shots categories without sacrificing the accuracy on many-shots, and outperforms BBN on many-shots by a large margin (more than 20% absolute increase).

Ablation study on distillation methods. Self-distillation step is optional but recommended if further improvements (0.4%$\sim$0.8% for most experiments) are desired. We apply distillation from a more powerful model with more experts into a model with fewer experts. A simple way to transfer knowledge is knowledge distillation (KD) (Hinton et al., 2015), which applies KD loss ($\mathcal{L}_{\text{KD}} = T^2 D_{\text{KL}}(\vec{l}_{\text{teacher}}/T, \vec{l}_{\text{expert}_i}/T)$) to match the distribution of logits of a teacher and a student. We found that for teacher model with more experts using smaller distillation loss factor gives better performance. We hypothesize that since we distill from the same teachers, giving large distillation factor prevents the branches from becoming as diversified as it is able to. We also explored other distillation methods, such as CRD (Tian et al., 2019), PKT (Passalis & Tefas, 2018), and SP (Tung

Table 5: **Comparison of different distillation methods.** We transfer from a model based on ResNet-32 with 6 experts to a model of the same type, except with fewer experts. We use CIFAR100-LT for the following comparison. No expert assignment module is used in the following experiments. Following the procedure for CRD (Tian et al., 2019), we also apply KD when we transfer from a teacher to students with other distillation methods.

Model Type	#expert	Distillation Method	Accuracy (%)
Teacher	6		49.7
Student	2	No Distillation	46.6
	2	KD (Hinton et al., 2015)	47.3
	2	CRD (Tian et al., 2019)	**47.5**
	2	PKT (Passalis & Tefas, 2018)	47.2
	2	SP (Tung & Mori, 2019)	47.2
	3	No Distillation	47.9
	3	KD (Hinton et al., 2015)	48.4
	3	CRD (Tian et al., 2019)	48.5
	3	PKT (Passalis & Tefas, 2018)	48.3
	3	SP (Tung & Mori, 2019)	**48.7**
	4	No Distillation	48.7
	4	KD (Hinton et al., 2015)	**49.3**
	4	CRD (Tian et al., 2019)	49.0
	4	PKT (Passalis & Tefas, 2018)	48.9
	4	SP (Tung & Mori, 2019)	49.0

& Mori, 2019), and compared the differences in Table 5. Although adding other methods along with KD may boost performance, the difference is small. Therefore, we opt for simplicity and use KD only unless otherwise noticed.

Detailed results for ImageNet-LT experiments. We list details of our ResNet-50 experiments in ImageNet-LT on Table 7. With 2 experts, we are able to achieve about 7% gain in accuracy with computational cost about 10% less than baseline. In contrast to previous methods that sacrifice many-shot accuracy to get few-shot accuracy, we improve on all three splits on ImageNet-LT. From 3 experts to 4 experts, we keep the same many-shot accuracy while increasing the few-shot accuracy, indicating that we are using the additional computational power to improve on the hardest part of the data rather than uniformly applying to all samples.

We also list our ResNet-10 and ResNeXt-50 experiments on Table 6 and 8, respectively, to compare against other works evaluated on these backbones. Our method also achieves lower computational cost and higher performance when compared to other methods.

As illustrated in Fig. 6, our approach provides a comprehensive treatment to all the many-shot, medium-shot and few-shot classes, achieving substantial improvements to current state-of-the-art on all aspects. Compared with cRT which reduces the performance on the many-shot classes, RIDE can achieves significantly better performance on the few-shot classes without impairing the many-shot classes. Similar observations can be obtained in the comparison with the state-of-the-art method BBN (Zhou et al., 2020) on iNaturalist.

Comparison with ensemble method. Since our method requires the joint decision from several experts, which raw ensembles also do, we also compare against ensembles of LDAM in Fig.7 on CIFAR100-LT. In the figure, even our method with 4 experts has less computational cost than the minimum computational cost for the ensemble of 2 LDAM models. This indicates that our model is much more efficient and powerful in terms of computational cost and accuracy than ensemble on long-tailed datasets.

t-SNE visualization. We also provide the t-SNE visualization of embedding space on CIFAR100-LT as in Fig. 8. Compared with the baseline method LDAM, the feature embedding of RIDE is more compact for both the head and tail classes and better separated from the neighboring classes. This greatly reduces the difficulty for the classifier to distinguish the tail category.

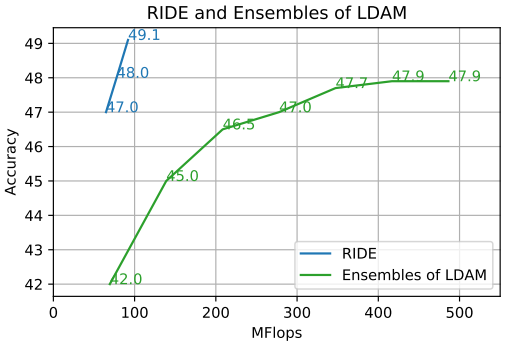


Figure 7: Comparison between our method and multiple LDAM models ensembled together. In the figure, ensembles of LDAM start from 1 ensemble (original LDAM) to 7 ensembles, and RIDE starts from 2 experts to 4 experts. Our method achieves higher accuracy with substantially less computational cost compared to ensemble method.

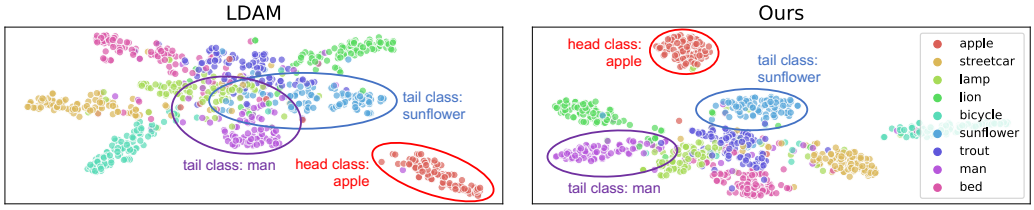


Figure 8: t-SNE visualization of LDAM's and our model's embedding space of CIFAR100-LT. The feature embedding of RIDE is more compact for both head and tail classes and better separated. This behavior greatly reduces the difficulty for the classifier to distinguish the tail category.

What if we apply RIDE to balanced datasets? We also conducted experiments on CIFAR100 to check if our method can achieve similar performance gains on balanced datasets. However, we only obtained an improvement of about 1%, which is much smaller than the improvements observed on the CIFAR100-LT. Compared with balanced datasets, long-tailed datasets can get more benefits from RIDE.

Table 6: Top-1 accuracy comparison with state-of-the-art methods on **ImageNet-LT** (Liu et al., 2019) with **ResNet-10**. Performance on Many-shot (>100), Medum-shot (≤100 & >20) and Few-shot (≤20) are also provided. Results marked with † are copied from (Liu et al., 2019). Results with ‡ are from (Xiang et al., 2020).

Methods	GFlops	Many	Medium	Few	Overall
Cross Entropy (CE) †	0.89 (1.0x)	40.9	10.7	0.4	20.9
Focal Loss † (Lin et al., 2017)	0.89 (1.0x)	36.4	29.9	16.0	30.5
Range Loss † (Zhang et al., 2017)	0.89 (1.0x)	35.8	30.3	17.6	30.7
Lifted Loss † (Oh Song et al., 2016)	0.89 (1.0x)	35.8	30.4	17.9	30.8
OLTR (Liu et al., 2019)	0.89 (1.0x)	43.2	35.1	18.5	35.6
LFME (Xiang et al., 2020)	-	47.0	37.9	19.2	38.8
Many-shot only ‡	-	59.3			
Medium-shot only ‡	-		35.9		
Few-shot only ‡	-			14.3	
RIDE (2 experts)	**0.85 (1.0x)**	57.5	40.8	26.9	45.3 **(+6.5)**
RIDE (3 experts)	0.97 (1.1x)	57.6	41.7	**28.0**	45.9 **(+7.1)**
RIDE (4 experts)	1.07 (1.2x)	**58.5**	**42.4**	27.7	**46.6** **(+7.8)**

Table 7: Top-1 accuracy comparison with state-of-the-art methods on **ImageNet-LT** (Liu et al., 2019) with **ResNet-50**. Performance on Many-shot (>100), Medum-shot (≤100 & >20) and Few-shot (≤20) are also provided. Results marked with † are copied from (Kang et al., 2020).

Methods	GFlops	Many	Medium	Few	Overall
Cross Entropy (CE) †	4.11 (1.0x)	64.0	33.8	5.8	41.6
NCM (Kang et al., 2020)	-	53.1	42.3	26.5	44.3
cRT (Kang et al., 2020)	4.11 (1.0x)	58.8	44.0	26.1	47.3
τ-norm (Kang et al., 2020)	4.11 (1.0x)	56.6	44.2	27.4	46.7
LWS (Kang et al., 2020)	4.11 (1.0x)	57.1	45.2	29.3	47.7
RIDE (2 experts)	**3.71 (0.9x)**	65.8	51.0	34.6	54.4 **(+6.7)**
RIDE (3 experts)	4.36 (1.1x)	**66.2**	51.7	34.9	54.9 **(+7.2)**
RIDE (4 experts)	5.15 (1.3x)	**66.2**	**52.3**	**36.5**	**55.4** **(+7.7)**

Table 8: Top-1 accuracy comparison with state-of-the-art methods on **ImageNet-LT** (Liu et al., 2019) with **ResNeXt-50**. Performance on Many-shot (>100), Medum-shot (≤100 & >20) and Few-shot (≤20) are also provided. Results marked with † are copied from (Kang et al., 2020).

Methods	GFlops	Many	Medium	Few	Overall
Cross Entropy (CE) †	4.26 (1.0x)	65.9	37.5	7.7	44.4
NCM (Kang et al., 2020)	-	56.6	45.3	28.1	47.3
cRT (Kang et al., 2020)	4.26 (1.0x)	61.8	46.2	27.4	49.6
τ-norm (Kang et al., 2020)	4.26 (1.0x)	59.1	46.9	30.7	49.4
LWS (Kang et al., 2020)	4.26 (1.0x)	60.2	47.2	30.3	49.9
RIDE (2 experts)	**3.92 (0.9x)**	67.6	52.5	35.0	55.9 **(+6.0)**
RIDE (3 experts)	4.69 (1.1x)	67.6	53.5	35.9	56.4 **(+6.5)**
RIDE (4 experts)	5.19 (1.2x)	**68.2**	**53.8**	**36.0**	**56.8** **(+6.9)**