DECOUPLED TRAINING FOR LONG-TAILED CLASSIFICATION WITH STOCHASTIC REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Decoupling representation learning and classifier learning has been shown to be effective in classification with long-tailed data. There are two main ingredients in constructing a decoupled learning scheme; 1) how to train the feature extractor for representation learning so that it provides generalizable representations and 2) how to re-train the classifier that constructs proper decision boundaries by handling class imbalances in long-tailed data. In this work, we first apply Stochastic Weight Averaging (SWA), an optimization technique for improving the generalization of deep neural networks, to obtain better generalizing feature extractors for long-tailed classification. We then propose a novel classifier re-training algorithm based on stochastic representation obtained from the SWA-Gaussian, a Gaussian perturbed SWA, and a self-distillation strategy that can harness the diverse stochastic representations based on uncertainty estimates to build more robust classifiers. Extensive experiments on CIFAR10/100-LT, ImageNet-LT, and iNaturalist-2018 benchmarks show that our proposed method improves upon previous methods both in terms of prediction accuracy and uncertainty estimation.

1 INTRODUCTION

While deep neural networks have achieved remarkable performance on various computer vision benchmarks (e.g., image classification (Russakovsky et al., 2015) and object detection (Lin et al., 2014)), there still are many challenges when it comes to applying them for real-world applications. One of such challenges is that the real-world classification data are *long-tailed* - the distribution of class frequencies exhibits a long tail, and many of the classes have only a few observations belonging to them. As a consequence, the class distribution of such data is extremely imbalanced, degrading the performance of a standard classification model trained with the balanced class assumption due to a paucity of samples from tail classes (Van Horn et al., 2018; Liu et al., 2019). Thus, it is worth exploring a novel technique dealing with long-tailed data for real-world deployments.

Decoupling (Kang et al., 2020) is a learning strategy proven to be effective for long-tailed data, where the representation learning via the feature extractor and classifier learning via the last classification layer are decoupled. Even for a classification network failing on long-tailed data, the representations obtained from the penultimate layer can be flexible and generalizable, provided that the feature extractor part is expressive enough (Donahue et al., 2014; Zeiler and Fergus, 2014; Girshick et al., 2014). The main motivation behind the decoupling is that the performance bottleneck of the long-tailed classification is mainly due to the improper decision boundaries set over the representation space. Based on this, Kang et al. (2020) has shown that a simple re-training of the last layer parameters significantly improves the performance.

The success of decoupling naturally motivates obtaining more informative representations from which the classifier re-training can benefit. To this end, we first investigate Stochastic Weight Averaging (SWA; Izmailov et al., 2018), which improves the generalization performance of deep neural networks by seeking flat minima in loss surfaces. Although SWA has been successful for various tasks involving deep neural networks (e.g., supervised learning (Izmailov et al., 2018), semi-supervised learning (Athiwaratkun et al., 2019) and domain generalization (Cha et al., 2021)), to the best of our knowledge, it has never been explored for long-tailed classification problems. In Section 3, we empirically show that a naïve application of SWA for long-tailed classification would fail

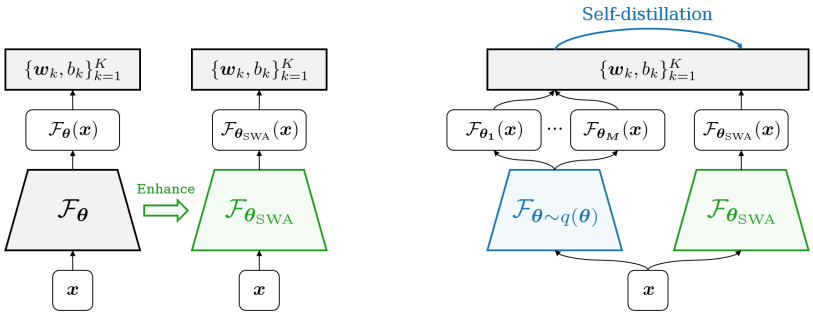

Figure 1: A schematic diagram depicting the overall composition of the paper. We first apply SWA to obtain more generalizing feature extractor (left; Section 3). We then propose a novel self-distillation strategy distilling SWAG teacher into SWA student to obtain more robust classifier (right; Section 4).

due to a similar bottleneck issue, but when combined with decoupling, SWA significantly improves the classification performance due to its property to obtain generalizable representations.

Confirming that SWA can benefit to long-tailed classification, we take a step further and propose a novel classifier re-training strategy. For this, we first obtain *stochastic representations*, the output of penultimate layers computed with multiple feature extractor parameters drawn from an approximate posterior, where we construct the approximate parameter with SWA-Gaussian (SWAG; Maddox et al., 2019). SWAG is a Bayesian extension of SWA, adding Gaussian noise to the parameter obtained from SWA to approximate posterior parameter uncertainty. In Section 3.2, we first show that the diverse stochastic representations obtained from SWAG samples well reflect the uncertainty of inputs. Hinging on this observation, we propose a novel self-distillation algorithm where the stochastic representations are used to construct an ensemble of *virtual* teachers, and the classifier re-training is formulated as a distillation (Hinton et al., 2015) from the virtual teacher.

Fig. 1 depicts the overall composition of this paper as a diagram. Using CIFAR10/100-LT (Cao et al., 2019), ImageNet-LT (Liu et al., 2019), and iNaturalist-2018 (Van Horn et al., 2018) benchmarks for long-tailed image classification, we empirically validate that our proposed method improves upon previous approaches both in terms of prediction accuracy and uncertainty estimation.

2 Preliminaries

2.1 Decoupled learning for long-tailed classification

Let $\mathcal{F}_{\boldsymbol{\theta}} : \mathbb{R}^D \to \mathbb{R}^L$ be a neural network parameterized by $\boldsymbol{\theta}$ that produces L-dimensional outputs for given D-dimensional inputs. For the K-way classification problem, an output from $\mathcal{F}_{\boldsymbol{\theta}}$ is first transformed into K-dimensional logits via a linear classification layer parameterized by $\boldsymbol{\phi} = (\boldsymbol{w}_k \in \mathbb{R}^L, b_k \in \mathbb{R})_{k=1}^K$, and then turned into a classification probability with the softmax function,

$$\boldsymbol{p}^{(k)}(\boldsymbol{x}; \boldsymbol{\Theta}) = \frac{\exp\left(\boldsymbol{w}_k^\top \mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{x}) + b_k\right)}{\sum_{j=1}^K \exp\left(\boldsymbol{w}_j^\top \mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{x}) + b_j\right)}, \quad \text{for } k = 1, ..., K, \tag{1}$$

where $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \boldsymbol{\phi})$ denotes a set of trainable parameters. Given a training set $\mathcal{D}$ consisting of pairs of input $\boldsymbol{x} \in \mathbb{R}^D$ and corresponding label $y \in \{1, \dots, K\}$, $\boldsymbol{\Theta}$ is trained to minimize the cross-entropy loss over $\mathcal{D}$,

$$\boldsymbol{\Theta}^* = (\boldsymbol{\theta}^*, \boldsymbol{\phi}^*) = \arg\min_{\boldsymbol{\Theta}} \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[-\log \boldsymbol{p}^{(y)}(\boldsymbol{x}; \boldsymbol{\Theta}) \right]. \tag{2}$$

Throughout the paper, we call $\mathcal{F}_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ as a *feature extractor*, and the last linear layer with parameters $\boldsymbol{\phi}$ as *classifier*. Also, we refer to a basic algorithm training the feature extractor and classifier together with Stochastic Gradient Descent (SGD) as SGD.

Previous works (Liu et al., 2019; Van Horn et al., 2018) have shown that the vanilla SGD suffers from the data paucity of tail classes when applied to long-tailed classification tasks. Notably, Kang

et al. (2020) showed that a simple re-training procedure on the classifier effectively resolves this problem. For instance, using pre-trained $\boldsymbol{\theta}^*$ from Eq. (2), we can re-train the classifier as

$$\boldsymbol{\phi}^{**} = \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\mathcal{D}_{\text{CB}}}} \left[-\log \boldsymbol{p}^{(y)}(\boldsymbol{x}; (\boldsymbol{\theta}^*, \boldsymbol{\phi})) \right], \tag{3}$$

where $\mathcal{D}_{\text{CB}}$ is the class-balanced training dataset, i.e., the probability of sampling a data $(\boldsymbol{x}, y)$ is given by $p_{\mathcal{D}_{\text{CB}}}((\boldsymbol{x}, y)) = 1/(K \times n_y)$, where n_y is the number of training examples for class y. This classifier Re-Training (cRT; Kang et al., 2020) method effectively improves the classification accuracy on tail classes without any changes in the feature extractor $\mathcal{F}_{\boldsymbol{\theta}^*}$.

2.2 Stochastic Weight Averaging (SWA)

Stochastic Weight Averaging (SWA; Izmailov et al., 2018) is an optimization method to improve generalization performance of deep neural networks. Given a loss function $\mathcal{L}(\Theta)$, the conventional SGD steps towards a local minima by following the gradient direction, where η denotes a step size,

$$\boldsymbol{\Theta}_t = \boldsymbol{\Theta}_{t-1} - \eta \nabla_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\Theta})|_{\boldsymbol{\Theta}=\boldsymbol{\Theta}_{t-1}}, \tag{4}$$

forming a parameter trajectory $\{\boldsymbol{\Theta}_t\}_{t\geq 1}$. SWA constructs a moving average of parameters for a periodically sampled subset of this trajectory, starting from $\boldsymbol{\Theta}_{\text{SWA}} = \mathbf{0}$,

$$\boldsymbol{\Theta}_{\text{SWA}} = (n\boldsymbol{\Theta}_{\text{SWA}} + \boldsymbol{\Theta}_n)/(n+1), \tag{5}$$

where $\boldsymbol{\Theta}_n$ is usually sampled at the end of every training epochs. This averaging in the weight space implicitly seeks flat minima in the loss surface, and thus enhances generalization. In practice, the averaging phase defined in Eq. (5) starts after the SGD trajectory falls into the basin of the loss function (e.g., after the 75% training epochs), and the learning rate during the averaging phase is set as high values to encourage exploration in the loss surface.

2.3 SWA-Gaussian for approximate Bayesian Inference

SWA-Gaussian (SWAG; Maddox et al., 2019) conducts Bayesian inference using a Gaussian approximation to the posterior distribution over the model parameters. With a slight abuse of notation writing the element-wise square as $\boldsymbol{\Theta}^2$, SWAG maintains the second moment for model parameters in addition to the first moment defined in Eq. (5), starting from $\boldsymbol{\Theta}'_{\text{SWA}} = \mathbf{0}$,

$$\boldsymbol{\Theta}'_{\text{SWA}} = (n\boldsymbol{\Theta}'_{\text{SWA}} + \boldsymbol{\Theta}_n^2)/(n+1), \tag{6}$$

to compute a diagonal covariance matrix $\boldsymbol{\Sigma}_{\text{SWAG}} = \text{diag}(\boldsymbol{\Theta}'_{\text{SWA}} - \boldsymbol{\Theta}^2_{\text{SWA}})$ approximating the sample covariance of parameters captured during SWA. The approximate posterior for the parameters is then constructed as Gaussian, $q(\boldsymbol{\Theta}) = \mathcal{N}(\boldsymbol{\Theta}; \boldsymbol{\Theta}_{\text{SWA}}, \boldsymbol{\Sigma}_{\text{SWAG}})$. As suggested in Maddox et al. (2019), one can also consider a higher-rank approximation for the covariance, but in this paper, we only consider the diagonal approximation.

3 Long-tailed classification with Stochastic Weight Averaging

After the success in ImageNet-1k classification (Russakovsky et al., 2015), SWA have been validated for various tasks, including semi-supervised learning (Athiwaratkun et al., 2019) and domain generalization (Cha et al., 2021). In this section, we study whether the success of SWA continues in the long-tailed classification (Section 3.1). Then we introduce the concept of stochastic representation constructed with SWAG, and empirically justify how such stochastic representations can benefit for the long-tailed classification (Section 3.2).

3.1 SWA needs the classifier retraining

Following Liu et al. (2019), we report classification accuracy (ACC) on three splits: Many (a set of classes each with over 100 training examples), Medium (a set of classes each with 20-100 training examples), and Few (a set of classes each with under 20 training examples). Table 1 compares SGD and SWA on large-scale benchmarks for long-tailed image classification, including ImageNet-LT and iNaturalist-2018. The experimental results '***Before applying cRT***' displayed in Table 1 show

Table 1: Comparison between SGD and SWA before and after applying cRT on benchmarks. Blue denotes a clear improvement of SWA over SGD, while red denotes deterioration.

	ImageNet-LT				iNaturalist-2018			
	Many	Medium	Few	All	Many	Medium	Few	All
Before applying cRT:								
SGD	66.84±0.26	40.78±0.24	12.05±0.23	**46.91**±0.22	76.31±0.52	67.89±0.18	62.24±0.17	**66.52**±0.05
SWA	67.71±0.11	40.74±0.15	11.01±0.10	**47.08**±0.12	77.26±0.25	68.23±0.25	61.87±0.13	**66.65**±0.10
After applying cRT:								
SGD	62.83±0.23	46.92±0.26	26.33±0.16	50.25±0.18	73.03±0.57	69.09±0.10	66.14±0.23	68.33±0.04
SWA	63.54±0.18	47.68±0.16	26.85±0.28	**50.95**±0.12	73.30±0.73	69.22±0.19	66.74±0.25	**68.66**±0.15

that SWA *does not* bring significant performance gain for long-tailed classification tasks, unlike the previous results on other tasks. Compared to the SGD, 1) SWA does not significantly improve performance, and 2) SWA even degrades performance for the 'Few' split, raising a question about the efficacy of SWA for the long-tailed classification. We diagnose the performance bottleneck of SWA from the perspective of decoupled learning. Specifically, we adopt cRT (Kang et al., 2020) method to verify whether the SWA inherently hinders the quality of the feature extractor. If SWA shows worse performance than SGD even after the classifier re-training, we can conclude that SWA is *not* preferable for representation learning for long-tailed classification. The results '***After applying cRT***' shown in Table 1 demonstrate the classification accuracy for SGD and SWA models after their classifiers are re-trained with Eq. (3). The result shows that SWA improves performance for all splits, indicating that SWA actually enhances the quality of the feature extractor, but the classification layer is acting as a bottleneck as in SGD, and this can be fixed with cRT.

3.2 Stochastic representations capture predictive uncertainties

Now let the feature extractor parameter be constructed from SWA procedure as $\boldsymbol{\theta}_{\text{SWA}}$ and the re-trained classifier parameter as $\boldsymbol{\phi}^*_{\text{SWA}}$. That is,

$$\boldsymbol{\phi}^*_{\text{SWA}} = \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\mathcal{D}_{\text{CB}}}} \Big[-\log \boldsymbol{p}^{(y)}(\boldsymbol{x}; (\boldsymbol{\theta}_{\text{SWA}}, \boldsymbol{\phi})) \Big]. \tag{7}$$

While the parameters $(\boldsymbol{\theta}_{\text{SWA}}, \boldsymbol{\phi}^*_{\text{SWA}})$ may generalize better than the one obtained from SGD, it is still a point estimate without fully capturing the uncertainty of the predictions. To overcome this limitation, we apply SWAG for the feature extractor parameters $\boldsymbol{\theta}$, approximating the posterior of $\boldsymbol{\theta}$ with a Gaussian distribution $q(\boldsymbol{\theta}|\mathcal{D}) := \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_{\text{SWA}}, \boldsymbol{\Sigma}_{\text{SWA}})$. Given $q(\boldsymbol{\theta}|\mathcal{D})$, a predictive distribution for an input $\boldsymbol{x}$ can be approximated as

$$\begin{aligned} \boldsymbol{p}^{(y)}(\boldsymbol{x}|\mathcal{D}; \boldsymbol{\phi}^*_{\text{SWA}}) &:= p(y=k|\boldsymbol{x}, \mathcal{D}; \boldsymbol{\phi}^*_{\text{SWA}}) \\ &\approx \int p^{(y)}(\boldsymbol{x}; (\boldsymbol{\theta}, \boldsymbol{\phi}^*_{\text{SWA}})) q(\boldsymbol{\theta}|\mathcal{D}) \mathrm{d}\boldsymbol{\theta} \approx \frac{1}{M} \sum_{m=1}^{M} p^{(y)}(\boldsymbol{x}; (\boldsymbol{\theta}_m, \boldsymbol{\phi}^*_{\text{SWA}})), \end{aligned} \tag{8}$$

where $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M \overset{\text{i.i.d.}}{\sim} q(\boldsymbol{\theta}|\mathcal{D})$. Here, we are computing the model average from multiple predictions evaluated with the multiple feature extractor parameters $(\boldsymbol{\theta}_m)_{m=1}^M$ to better capture epistemic uncertainty. During that process, we implicitly compute the *stochastic representations* of $\boldsymbol{x}$, that is,

$$\mathcal{F}_m(\boldsymbol{x}) := \mathcal{F}_{\boldsymbol{\theta}_m}(\boldsymbol{x}) \text{ for } m = 1, \ldots, M. \tag{9}$$

We hypothesize that these stochastic representations reflect the epistemic uncertainty of $\boldsymbol{x}$, so it is important to consider them for the classifier re-training stage. For instance, if $\boldsymbol{x}$ is a hard example (e.g., a sample from tail classes) that is likely to be misclassified, the corresponding stochastic representations are expected to produce predictive distribution having high variances.

Measuring per-instance dispersion. We empirically validate our hypothesis on stochastic representations by measuring their dispersion over different inputs. The average cosine distance of the set elements to the set centroid in the L-dimensional representation space can quantify such dispersion[1]. To be concrete, we consider the following *per-instance dispersion* for each instance $\boldsymbol{x}$ in the

[1]We use the cosine distance instead of Euclidean distance since it is a more reasonable choice for the inner-product-based representation space trained with Eq. (1).

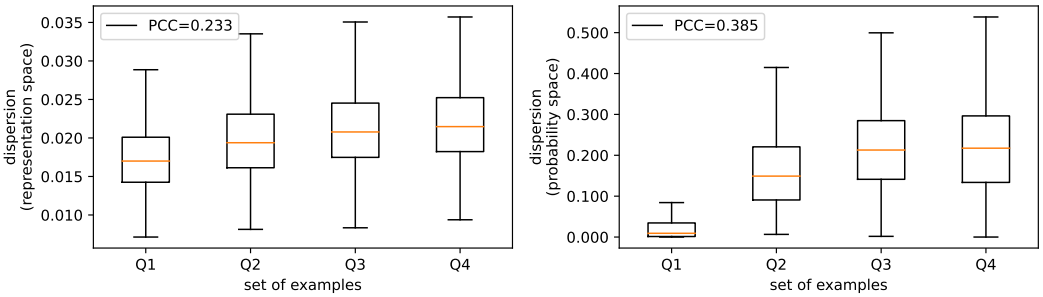

Figure 2: Box-and-whisker plots for dispersion values in four different groups of examples. We also compute the Pearson Correlation Coefficient (PCC) value to clarify the positive correlation between NLL and dispersion.

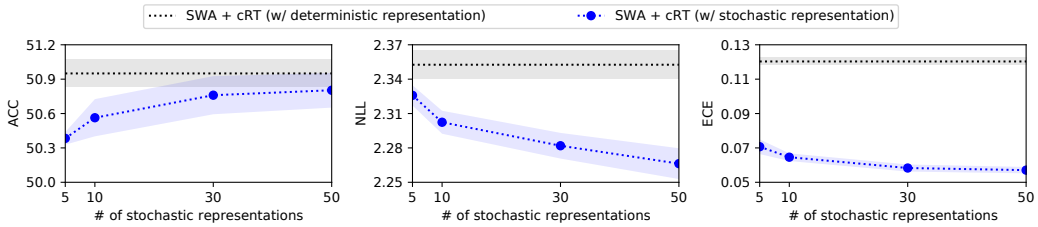

Figure 3: Results for ensemble of stochastic representations over re-trained decision boundaries. Along with classification accuracy (ACC; left), we plot uncertainty estimates, including negative log-likelihood (NLL; center) and expected calibration error (ECE; right). We evaluate models on the test split of ImageNet-LT and colorize standard deviations over 4 seeds.

representation space,

$$\text{Dispersion}\left(\boldsymbol{x}; \{\boldsymbol{\theta}_m\}_{m=1}^M\right) := \frac{1}{M}\sum_{m=1}^{M}\left[1 - \frac{\overline{\mathcal{F}}(\boldsymbol{x})^\top \mathcal{F}_{\boldsymbol{\theta}_m}(\boldsymbol{x})}{\left\|\overline{\mathcal{F}}(\boldsymbol{x})\right\|_2 \left\|\mathcal{F}_{\boldsymbol{\theta}_m}(\boldsymbol{x})\right\|_2}\right], \tag{10}$$

where $\overline{\mathcal{F}}(\boldsymbol{x}) := \sum_{m=1}^{M} \mathcal{F}_{\boldsymbol{\theta}_m}(\boldsymbol{x})/M$ is the centroid of M stochastic representations $\{\mathcal{F}_{\boldsymbol{\theta}_m}(\boldsymbol{x})\}_{m=1}^M$ for an input $\boldsymbol{x}$. Furthermore, we also measure the Jensen-Shannon Divergence (JSD) for a set of predictions $\{\boldsymbol{p}(\boldsymbol{x}; (\boldsymbol{\theta}_m, \boldsymbol{\phi}^*_{\text{SWA}}))\}_{m=1}^M$,

$$\text{Dispersion}\left(\boldsymbol{x}; \{(\boldsymbol{\theta}_m, \boldsymbol{\phi}_{\text{SWA}^*})\}_{m=1}^M\right) := \text{JSD}\left(\{\boldsymbol{p}(\boldsymbol{x}; (\boldsymbol{\theta}_m, \boldsymbol{\phi}^*_{\text{SWA}}))\}_{m=1}^M\right), \tag{11}$$

to quantify the *per-instance dispersion* for each instance $\boldsymbol{x}$ in the probability space.

Fig. 2 shows the box-and-whisker plots for dispersion values over 20,000 validation examples from ImageNet-LT. We split them into four disjoint groups consisting of 5,000 instances in each group based on the Negative Log-Likelihood (NLL) values: Q1 (a set of instances having NLLs lower than the first quartile), Q2 (a set of instances having NLLs in the range between the first and second quartiles), Q3 (a set of instances having NLLs in the range between the second and third quartiles), and Q4 (a set of instances having NLLs higher than the third quartile). We confirm that there exists a positive correlation between NLL and dispersion (0.233 for Eq. (10) and 0.385 for Eq. (11)), that is, a hard example (i.e., higher NLL) tends to have more dispersed stochastic representations (i.e., higher dispersion). The higher correlation in Eq. (11) compared to Eq. (10) motivates us to harness the diversity in the probability space instead of the representation space. Empirical results in Section 6.1 will show this indeed leads to improvements.

Ensembling predictions with stochastic representations. We further provide an empirical evidence showing that the uncertainty captured by the stochastic representations can be harnessed by the classifier for robust classification. Specifically, we compute the classification probabilities ensembled over stochastic representations using (8). Note that the classifier parameters $\boldsymbol{\phi}^*_{\text{SWA}}$ is re-trained with $\boldsymbol{\theta}_{\text{SWA}}$ with (7), not with the parameters $(\boldsymbol{\theta}_m)_{m=1}^M$, so in principle, there is no guarantee

that ϕ^*_{SWA} is compatible with the stochastic representations computed from $(\boldsymbol{\theta}_m)_{m=1}^M$. Still, if the stochastic representations properly capture the uncertainty of $\boldsymbol{x}$ in the feature space, the ensembled classifier would provide well-calibrated predictions.

Fig. 3 demonstrates how classification accuracy and uncertainty estimates, including NLL and Expected Calibration Error (ECE), vary along with the number of stochastic representations for ensembling defined in Eq. (8). While the classification accuracy remains at a similar level, uncertainty estimates improve as the number of stochastic representations increases, indicating that the stochastic representations indeed captures the uncertainty of inputs that are helpful for robust predictions.

We would like to emphasize that the classifier Eq. (8) is a proof-of-concept model to validate our hypothesis, and there still is a room for improvement. First of all, as we mentioned earlier, the classifier parameters ϕ^*_{SWA} needs to be retrained according to the stochastic representations. Second, even with such re-training, the current form requires multiple forward pass through the feature extractor (with multiple $\boldsymbol{\theta}_m$), incurring undesirably heavy inference cost. In Section 4, we propose a novel re-training algorithm to resolve these issues.

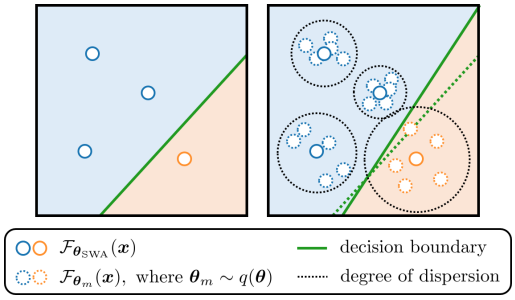

Figure 4: A schematic diagram depicting two-dimensional representation space after the first representation learning stage. The vanilla approach for the second classifier learning stage re-trains the decision boundary between two classes using deterministic representations (left). Our proposed methods consider stochastic representations having different degrees of dispersion for each input to build more robust decision boundary (right).

4 Decoupled Learning with Stochastic Representation

4.1 Re-training Classifier with Stochastic Representations

A straightforward way to re-training classifier with stochastic representations would be minimizing

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\mathcal{D}_{\text{CB}}}} \left[\mathbb{E}_{\boldsymbol{\theta}\sim q(\boldsymbol{\theta}|\mathcal{D})} [-\log p^{(y)}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})] \right] \approx \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\mathcal{D}_{\text{CB}}}} \left[\hat{\mathcal{L}}_{\text{CE}}(\boldsymbol{x}, y) \right], \quad (12)$$

where $\boldsymbol{p}^{(y)}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}) := \boldsymbol{p}^{(y)}(\boldsymbol{x}; (\boldsymbol{\theta}, \boldsymbol{\phi}))$, $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M \overset{\text{i.i.d.}}{\sim} q(\boldsymbol{\theta}|\mathcal{D})$, and

$$\hat{\mathcal{L}}_{\text{CE}}(\boldsymbol{x}, y) := -\frac{1}{M} \sum_{m=1}^{M} \log \boldsymbol{p}^{(y)}_{\boldsymbol{\theta}_m,\boldsymbol{\phi}}(\boldsymbol{x}). \quad (13)$$

Fig. 4 depicts a schematic diagram for two-dimensional representation space during the classifier re-training stage. While learning from a deterministic feature extractor (i.e., minimizing Eq. (7)) computes the deterministic representations, our proposed method minimizing Eq. (12) builds more robust decision boundaries since it accounts for predictive uncertainties estimated from the stochastic representations.

4.2 Self-distillation with Stochastic Representation

While the re-training with (12) helps build a robust classifier, there still is room for improvement. After all, what we are ultimately interested in is the uncertainty in a set of predictions $\{\boldsymbol{p}_{\boldsymbol{\theta}_m,\boldsymbol{\phi}}(\boldsymbol{x})\}_{m=1}^M$, not the stochastic representations $\{\mathcal{F}_{\boldsymbol{\theta}_m}(\boldsymbol{x})\}_{m=1}^M$ themselves. The objective (12) learns the classifier parameters $\boldsymbol{\phi}$ with the *mean* loss (13), so it does not consider the *diversities* of the predictions made by different stochastic representations. Moreover, as we pointed out earlier, in the current form, we need M forward passes through the feature extractor with $(\boldsymbol{\theta}_m)_{m=1}^M$ to make a prediction. To resolve these issues, we present a self-distillation algorithm that can enhance the robustness of the classifier by transferring diversities in the multiple predictions made with stochastic representations reducing the inference cost to be the same as a single model.

Setting up teachers and a student. We first set a set of teachers constructed from the set of stochastic representations. Formally, the mth teacher $\mathcal{T}_m$ produces a prediction probability for an input $\boldsymbol{x}$ as $\boldsymbol{p}_{\mathcal{T}_m}(\boldsymbol{x}) := \boldsymbol{p}_{\boldsymbol{\theta}_m,\boldsymbol{\phi}}(\boldsymbol{x})$ where $\boldsymbol{\theta}_m \sim q(\boldsymbol{\theta}|\mathcal{D})$. For a student, we first fix the feature extractor parameter as $\boldsymbol{\theta}_{\text{SWA}}$, which is a natural choice considering that the mean of the distribution $q(\boldsymbol{\theta}|\mathcal{D})$ is $\boldsymbol{\theta}_{\text{SWA}}$. The goal is to learn the classifier parameters $\boldsymbol{\phi}$ such that the student prediction $\boldsymbol{p}_{\boldsymbol{\theta}_{\text{SWA}},\boldsymbol{\phi}}(\boldsymbol{x})$ can maximally absorb the diversities in the teacher predictions $(\boldsymbol{p}_{\mathcal{T}_m}(\boldsymbol{x}))_{m=1}^{M}$.

Distillation objective. Following Ryabinin et al. (2021), instead of directly distilling from the *mean* of ensemble predictions $\boldsymbol{p}_{\mathcal{T}}(\boldsymbol{x}) := \sum_m \boldsymbol{p}_{\mathcal{T}_m}(\boldsymbol{x})/M$ as in the original knowledge distillation (Hinton et al., 2015), we distill from the *distribution* of the ensembled predictions. Specifically, we assume that the teacher prediction probabilities $(\boldsymbol{p}_{\mathcal{T}_m}(\boldsymbol{x}))_{m=1}^{M}$ are Dirichlet distributed with parameter $\boldsymbol{\beta}(\boldsymbol{x})$, that is,

$$\boldsymbol{p}_{\mathcal{T}_1}(\boldsymbol{x}), \ldots, \boldsymbol{p}_{\mathcal{T}_M}(\boldsymbol{x}) \overset{\text{i.i.d.}}{\sim} \text{Dir}(\boldsymbol{\beta}(\boldsymbol{x})). \tag{14}$$

The parameter can be estimated via the maximum likelihood procedure whose solution is given in a closed-form (Minka, 2000):

$$\boldsymbol{\beta}^{(k)}(\boldsymbol{x}) \coloneqq \boldsymbol{p}_{\mathcal{T}}^{(k)}(\boldsymbol{x}) \frac{(K-1)/2}{\sum_{j=1}^{K}\left[\boldsymbol{p}_{\mathcal{T}}^{(j)}(\boldsymbol{x})\left(\log \boldsymbol{p}_{\mathcal{T}}^{(j)}(\boldsymbol{x}) - \frac{1}{M}\sum_{m=1}^{M}\log \boldsymbol{p}_{\mathcal{T}_m}^{(j)}(\boldsymbol{x})\right)\right]}, \tag{15}$$

for $k = 1, ..., K$. Having computed $\boldsymbol{\beta}(\boldsymbol{x})$, we assume that the output probabilities from a student model (our original model to be re-trained for classifier parameters) are also Dirichlet distributed with parameter $\boldsymbol{\alpha}(\boldsymbol{x})$. Following Ryabinin et al. (2021), we constraint all components in $\boldsymbol{\beta}$ to be greater than 1 (i.e., $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \mathbf{1}$) and minimize the reverse KL divergence $D_{\text{KL}}[\text{Dir}(\boldsymbol{\alpha})\|\text{Dir}(\boldsymbol{\beta})]$ for stable training of $\boldsymbol{\alpha}$. Expanding the KL divergence, we obtain the distillation loss as

$$\hat{\mathcal{L}}_{\text{KD}}(\boldsymbol{x}, y) = \mathbb{E}_{\boldsymbol{p}\sim\text{Dir}(\boldsymbol{\alpha})}\left[-\sum_{k=1}^{K}\boldsymbol{p}_{\mathcal{T}}^{(k)}(\boldsymbol{x})\log \boldsymbol{p}^{(k)}\right] + \frac{1}{\sum_{j=1}^{K}\boldsymbol{\beta}^{(j)}} D_{\text{KL}}\left[\text{Dir}(\boldsymbol{\alpha})\|\text{Dir}(\mathbf{1})\right], \tag{16}$$

where we compute $\boldsymbol{\alpha}(\boldsymbol{x}) \in \mathbb{R}_{+}^{K}$ for $\boldsymbol{\phi} = (\boldsymbol{w}_k, b_k)_{k=1}^{K}$ as

$$\boldsymbol{\alpha}^{(k)}(\boldsymbol{x}) \coloneqq \exp\left(\boldsymbol{w}_k^{\top}\mathcal{F}_{\boldsymbol{\theta}_{\text{SWA}}}(\boldsymbol{x}) + b_k\right) \quad \text{for } k = 1, ..., K. \tag{17}$$

After learning $\boldsymbol{\alpha}(\boldsymbol{x})$, we can take the mean of the student Dirichlet distribution as a representative classification probability for $\boldsymbol{x}$. By the property of the softmax, this actually coincides with the $\boldsymbol{p}_{\boldsymbol{\theta}_{\text{SWA}},\boldsymbol{\phi}}(\boldsymbol{x})$. That is,

$$\mathbb{E}_{\boldsymbol{p}\sim\text{Dir}(\boldsymbol{\alpha}(\boldsymbol{x}))}\left[\boldsymbol{p}^{(k)}\right] = \frac{\boldsymbol{\alpha}^{(k)}(\boldsymbol{x})}{\sum_{j=1}^{K}\boldsymbol{\alpha}^{(j)}(\boldsymbol{x})} = \boldsymbol{p}_{\boldsymbol{\theta}_{\text{SWA}},\boldsymbol{\phi}}^{(k)}(\boldsymbol{x}). \tag{18}$$

The final version of our algorithm combines the two loss terms $\hat{\mathcal{L}}_{\text{CE}}$ and $\hat{\mathcal{L}}_{\text{KD}}$ for the classifier re-training (see Appendix A.4 for more details). While the first term builds decision boundaries over the stochastic representations, the second term further adjusts decision boundaries by distilling diverse predictions from the stochastic representations. We name this procedure as a *self-distillation* in the sense that the model distills probabilistic outputs within the network itself (Zhang et al., 2019). We empirically validate the effectiveness of the self-distillation in Section 6.

5 Related Works

Decoupled learning. Recent works have shown that the performance bottleneck of deep neural classifiers on long-tailed datasets is improper decision boundaries (Kang et al., 2020; Zhang et al., 2021). The vanilla training with instance-balanced sampling gives generalizable representations (Kang et al., 2020), and thus a simple adjusting strategy for the classifier can alleviate such bottleneck. For instance, re-training the classifier from scratch or normalizing the classifier weights with class-balanced sampling (Kang et al., 2020), adjusting the classifier biases with the empirical class frequencies on the training dataset (Menon et al., 2021), and training additional module which performs input-dependent adjustment of the original classifier (Zhang et al., 2021).

Table 2: Ablation studies of proposed methods on ImageNet-LT: classification accuracy (ACC), negative log-likelihood (NLL), and expected calibration error (ECE). These results are with Class Balanced Sampling (CBS); refer to Appendix B.1 the results for the other balancing strategies.

Method	Note	ACC (↑)	NLL (↓)	ECE (↓)
SGD w/ classifier re-training	Section 2.1	$50.25_{\pm 0.18}$	$2.364_{\pm 0.008}$	$0.110_{\pm 0.001}$
+ (a) introducing SWA for the representation learning	Section 3	$50.95_{\pm 0.12}$	$2.353_{\pm 0.012}$	$0.120_{\pm 0.002}$
+ (b) classifier re-training w/ stochastic representation	Section 4.1	$51.33_{\pm 0.17}$	$2.340_{\pm 0.012}$	$0.125_{\pm 0.003}$
+ (c) classifier re-training w/ self-distillation	Section 4.2	$\mathbf{51.66}_{\pm 0.13}$	$\mathbf{2.203}_{\pm 0.009}$	$\mathbf{0.074}_{\pm 0.002}$

Knowledge distillation. The seminal work of Hinton et al. (2015) has shown that distilling knowledge in deep neural networks is an effective way to obtain better generalizing models. One of the common practices in knowledge distillation is employing ensembles as a teacher model (Malinin et al., 2020; Ryabinin et al., 2021) based on the superior performance of the ensemble of deep neural networks (Lakshminarayanan et al., 2017; Ovadia et al., 2019), and several works already applied this to long-tailed recognition (Iscen et al., 2021; Wang et al., 2021). Nevertheless, we would like to clarify that there is a clear difference from them in that our method does not require a costly teacher model (i.e., we do not require an independently trained teacher model).

6 Experiments

We present extensive experimental results on long-tailed image classification benchmarks for the family of residual networks (He et al., 2016), i.e., ResNet-32 on CIFAR10/100-LT (Cao et al., 2019) and ResNet-50 on ImageNet-LT (Liu et al., 2019) and iNaturalist-2018 (Van Horn et al., 2018). Refer to Appendices C.1 and C.2 for more details on each setup. While some previous literature only reports classification accuracy (ACC), we also provide Negative Log-Likelihood (NLL) and Expected Calibration Error (ECE) that further evaluate the calibration of classification models. See Appendix A.3 for definitions of each metric. Unless specified, we report numbers in Avg.$\pm$std. over four random seeds.

6.1 Ablation studies of proposed methods

In this section, we empirically verify that our proposed method progressively improves upon the previous baseline. More precisely, we test the following step-by-step; (a) introducing SWA for the representation learning from Section 3, (b) introducing stochastic representation for the classifier re-training from Section 4.1, and (c) introducing self-distillation strategy from Section 4.2.

Here, we consider the following balancing strategies during the classifier re-training; 1) Class-Balanced Sampling (CBS; Kang et al., 2020), 2) Generalized Re-Weighting (GRW; Zhang et al., 2021), and 3) Logit Adjustment (LA; Menon et al., 2021). Such strategies to overcome the difficulty of imbalanced data distribution are essential to re-train proper decision boundaries in the decoupled learning scheme. Refer to Appendix A.1 for more details on balancing strategies.

Table 2 shows the evaluation results for each step on ImageNet-LT. To summarize, (a) the results displayed in the first and second rows show that SWA improves the SGD baseline with a decoupling scheme, as we discussed before in Section 3.1. (b) Moreover, the results displayed in the second and third rows show that using stochastic representations to re-train the classifier improves the classification accuracy. It indicates that the stochastic representations contribute to building more robust decision boundaries, as depicted in Fig. 4. However, our experimental results up to this point show that the improvements in uncertainty estimates are not significant. (c) Finally, our proposed approach in Section 4.2 significantly outperforms previous baselines for every metric we measured. Notably, the improvement consistently appears for all balancing strategies we considered (see Appendix B.1 for the full results). In particular, definite improvement in uncertainty estimates confirms the effectiveness of the proposed self-distillation method.

6.2 Results on image classification tasks

We compare our approach to the existing methods for long-tailed learning; cRT (Kang et al., 2020), LWS (Kang et al., 2020), LA (Menon et al., 2021), and DisAlign (Zhang et al., 2021). See Ap-

Table 3: Results on ImageNet-LT and iNaturalist-2018: classification accuracy (ACC), negative log-likelihood (NLL), and expected calibration error (ECE). Full results are available in Appendix B.1.

Method	ImageNet-LT			iNaturalist-2018		
	ACC (↑)	NLL (↓)	ECE (↓)	ACC (↑)	NLL (↓)	ECE (↓)
SWA	47.08±0.12	2.631±0.009	0.187±0.002	66.65±0.10	1.568±0.005	0.071±0.001
+ cRT (Kang et al., 2020)	50.95±0.12	2.353±0.012	0.120±0.002	68.66±0.15	1.546±0.002	0.061±0.002
+ LWS (Kang et al., 2020)	51.60±0.10	2.189±0.007	0.077±0.002	70.53±0.09	1.370±0.002	0.049±0.001
+ LA (Menon et al., 2021)	51.62±0.05	2.206±0.009	0.077±0.002	69.63±0.20	1.466±0.003	0.038±0.001
+ DisAlign (Zhang et al., 2021)	**52.18**±0.11	2.673±0.014	0.215±0.002	**70.81**±0.10	1.410±0.003	0.076±0.000
+ SRepr (ours)	**52.12**±0.06	**2.130**±0.006	**0.037**±0.001	**70.79**±0.17	**1.353**±0.002	**0.036**±0.002

Table 4: Further comparisons in classification accuracy (ACC) with state-of-the-art methods on ImageNet-LT. Results for the baselines came from the corresponding paper. †This actually requires training cost of 600 epochs since rwSAM doubles forward and backward passes for each iteration.

Method	Epoch	Network	Training details	ACC (↑)
KCL (Kang et al., 2021)	200	ResNet-50	-	51.5
SWA + SRepr (ours)	200	ResNet-50	-	**53.8**
MiSLAS (Zhong et al., 2021)	180	ResNet-50	mixup (Zhang et al., 2018)	52.7
SWA + SRepr (ours)	180	ResNet-50	mixup (Zhang et al., 2018)	**53.9**
LADE (Hong et al., 2021)	180	ResNeXt-50 (Xie et al., 2017)	-	53.0
SWA + SRepr (ours)	180	ResNeXt-50 (Xie et al., 2017)	-	**54.6**
Liu et al. (2021)	300	ResNet-50	MoCo v2 (He et al., 2020)	55.0
SWA + SRepr (ours)	300	ResNet-50	mixup (Zhang et al., 2018)	**55.1**
Liu et al. (2021)	300†	ResNet-50	MoCo v2 (He et al., 2020) + rwSAM (Foret et al., 2021)	55.5
SWA + SRepr (ours)	400	ResNet-50	mixup (Zhang et al., 2018)	**55.7**

pendices A.1 and A.2 for more details on existing methods. Since the representation learning using SWA, we introduced in Section 3, is also compatible with previous classifier re-training methods, we hereby report the results built upon this for a fair comparison.

Table 3 presents the evaluation results on ImageNet-LT and iNaturalist-2018. **SWA + SRepr** denotes the final version of our proposed method, previously discussed in Section 6.1, i.e., self-distillation using stochastic representation and balanced by logit adjustment. It demonstrates that the proposed SRepr outperforms baselines, even if existing methods strengthen with SWA. While DisAlign, improved by our proposed SWA, achieves competitive accuracy, SRepr still shows better uncertainty estimates. Refer to Appendix B.1 for a full table, including classification accuracy for each split (i.e., Many, Medium, and Few) and baseline results without applying SWA. Appendix B.2 also provides the results on CIFAR10/100-LT, which show the same tendency. Besides, Appendix B.5 provides further analysis of ours, which validates the effectiveness of the proposed method.

Further comparisons with state-of-the-art methods. One can claim the performance gap between ours and the state-of-the-art methods. This issue appears due to the shorter training epochs (i.e., 100 training epochs) and the vanilla data augmentation strategy (i.e., random resized cropping and horizontal flipping) in our experimental setups. Table 4 further clarifies that our proposed approach outperforms the state-of-the-art methods (Kang et al., 2021; Zhong et al., 2021; Hong et al., 2021; Liu et al., 2021) when the training costs get in line (see Appendix B.3 for more details). Moreover, Appendix B.4 further verifies the compatibility between ours and the existing state-of-the-art methods (Wang et al., 2021; Park et al., 2022; Zhang et al., 2022; Zhu et al., 2022), which clarifies ours can be combined with existing algorithms taking different approaches.

7 Conclusion

In this paper, we proposed a simple yet effective classifier re-training strategy for long-tailed learning in decoupled learning scheme. We first showed that successful representation learning is achievable by SWA without any complex training methods. To the best of our knowledge, this is the first attempt to introduce SWA into long-tailed learning. While just combining existing classifier re-training methods with the representation learned by SWA shows better performance than with the vanilla SGD, we further proposed a novel self-distillation strategy that significantly improves uncertainty estimates of the final classifier.

Ethic and reproducibility statements. For societal impacts, there can be potential negative impacts associated with face recognition (e.g., privacy threats) since face data often exhibit long-tailed distribution over entities (Zhang et al., 2017). For reproducibility, Appendix C contains all the experimental setup including datasets and hyperparameters.

REFERENCES

S. Alshammari, Y.-X. Wang, D. Ramanan, and S. Kong. Long-tailed recognition via weight balancing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 19, 21

B. Athiwaratkun, M. Finzi, P. Izmailov, and A. G. Wilson. There are many consistent explanations of unlabeled data: Why you should average. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 3

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax. 20

K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019. 2, 8, 21

J. Cha, S. Chun, K. Lee, H.-C. Cho, S. Park, Y. Lee, and S. Park. Swad: Domain generalization by seeking flat minima. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, 2021. 1, 3

J. Cui, Z. Zhong, S. Liu, B. Yu, and J. Jia. Parametric contrastive learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 17

J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of The 31st International Conference on Machine Learning (ICML 2014)*, 2014. 1

P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations (ICLR)*, 2021. 9

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 1

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017. 19

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 8, 21

K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 9

J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee. Flax: A neural network library and ecosystem for JAX, 2020. URL http://github.com/google/flax. 20

M. Hessel, D. Budden, F. Viola, M. Rosca, E. Sezener, and T. Hennigan. Optax: composable gradient transformation and optimisation, in jax!, 2020. URL http://github.com/deepmind/optax. 20

G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *Deep Learning and Representation Learning Workshop, NIPS 2014*, 2015. 2, 7, 8

Y. Hong, S. Han, K. Choi, S. Seo, B. Kim, and B. Chang. Disentangling label distribution for long-tailed visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 9, 17

A. Iscen, A. Araujo, B. Gong, and C. Schmid. Class-balanced distillation for long-tailed visual recognition. In *The British Machine Vision Conference (BMVC)*, 2021. 8

P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 2018. 1, 3, 22

B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 2, 3, 4, 7, 8, 9, 14, 16, 17, 18, 19

B. Kang, Y. Li, S. Xie, Z. Yuan, and J. Feng. Exploring balanced feature spaces for representation learning. In *International Conference on Learning Representations (ICLR)*, 2021. 9

A. Krogh and J. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems 4 (NIPS 1991)*, 1991. 21

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, 2017. 8, 18

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, 2014. 1

H. Liu, J. Z. HaoChen, A. Gaidon, and T. Ma. Self-supervised learning is more robust to dataset imbalance. In *International Conference on Learning Representations (ICLR)*, 2021. 9

Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu. Large-scale long-tailed recognition in an open world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 3, 8, 16, 20

W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019. 2, 3

A. Malinin, B. Mlodozeniec, and M. J. F. Gales. Ensemble distribution distillation. In *International Conference on Learning Representations (ICLR)*, 2020. 8

A. K. Menon, S. Jayasumana, A. S. Rawat, H. Jain, A. Veit, and S. Kumar. Long-tail learning via logit adjustment. In *International Conference on Learning Representations (ICLR)*, 2021. 7, 8, 9, 14, 16, 17, 18

T. Minka. Estimating a dirichlet distribution, 2000. 7

Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019. 8, 18

S. Park, Y. Hong, B. Heo, S. Yun, and J. Y. Choi. The majority can help the minority: Context-rich minority oversampling for long-tailed classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 9, 18

J. Ren, C. Yu, X. Ma, H. Zhao, S. Yi, et al. Balanced meta-softmax for long-tailed visual recognition. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020. 19

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015. 1, 3, 21

M. Ryabinin, A. Malinin, and M. Gales. Scaling ensemble distribution distillation to many classes with proxy targets. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, 2021. 7, 8

G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 8, 21

X. Wang, L. Lian, Z. Miao, Z. Liu, and S. Yu. Long-tailed recognition by routing diverse distribution-aware experts. In *International Conference on Learning Representations (ICLR)*, 2021. 8, 9, 17, 18

A. G. Wilson and P. Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020. 18

L. Xiang, G. Ding, and J. Han. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *European Conference on Computer Vision*, 2020. 17

S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 9, 17

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 2014. 1

H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018. 9, 17

L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 7

S. Zhang, Z. Li, S. Yan, X. He, and J. Sun. Distribution alignment: A unified framework for long-tail visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 7, 8, 9, 14, 15, 16, 17, 18

X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao. Range loss for deep face recognition with long-tailed training data. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 10

Y. Zhang, B. Hooi, L. Hong, and J. Feng. Self-supervised aggregation of diverse experts for test-agnostic long-tailed recognition. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022. 9, 18

Z. Zhong, J. Cui, S. Liu, and J. Jia. Improving calibration for long-tailed recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 9, 17

B. Zhou, Q. Cui, X.-S. Wei, and Z.-M. Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 17

J. Zhu, Z. Wang, J. Chen, Y.-P. P. Chen, and Y.-G. Jiang. Balanced contrastive learning for long-tailed visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 9, 17, 18

A Supplementary Materials

A.1 Balancing strategies for re-training classifier

In Section 6.1, we considered the following balancing strategies for the classifier re-training stage. Such strategies for re-balancing the training data having long-tailed distribution over classes is essential to construct proper decision boundaries over the representation space.

- **Class-Balanced Sampling (CBS)** is the most straightforward way to re-balance long-tailed data distribution by sampling. The probability of sampling a training example from class k is $1/K$, where K is the number of training classes. More precisely, the softmax cross-entropy loss over the class-balanced training dataset $\mathcal{D}_{\text{CB}}$ is

$$\mathbb{E}_{(\boldsymbol{x},y)\sim p_{\mathcal{D}_{\text{CB}}}}\left[-\log\frac{\exp\left(\boldsymbol{w}_y^\top\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{x})+b_y\right)}{\sum_{j=1}^K\exp\left(\boldsymbol{w}_j^\top\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{x})+b_j\right)}\right], \tag{19}$$

  where the probability of sampling a data $(\boldsymbol{x},y)$ is given by $p_{\mathcal{D}_{\text{CB}}}((\boldsymbol{x},y))=1/(K\times n_y)$ and n_y denotes the number of training examples for class y.
- **Generalized Re-Weighting (GRW)** performs loss re-weighting using the empirical class frequencies $\{\pi_k\}_{k=1}^K$ on the training dataset (Zhang et al., 2021). More precisely, the re-weighted version of the softmax cross-entropy loss over the training dataset $\mathcal{D}$ is

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[-\frac{(1/\pi_y)^\rho}{\sum_{j=1}^K(1/\pi_j)^\rho}\log\frac{\exp\left(\boldsymbol{w}_y^\top\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{x})+b_y\right)}{\sum_{j=1}^K\exp\left(\boldsymbol{w}_j^\top\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{x})+b_j\right)}\right], \tag{20}$$

  where $\rho\geq 0$ is a hyper-parameter that controls the scale of per-class weighting coefficients, e.g., it reduces to the instance-balanced re-weighting with $\rho=0.0$, and to the class-balanced re-weighting with $\rho=1.0$. We use $\rho=1.0$ throughout the experiments.
- **Logit Adjustment (LA)** is a simple but efficient way to minimize the balanced error (i.e., average of per-class error rates) using the empirical class frequencies $\{\pi_k\}_{k=1}^K$ on the training dataset (Menon et al., 2021). More precisely, the logit adjusted version of the softmax cross-entropy loss over the training dataset $\mathcal{D}$ is

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[-\log\frac{\exp\left(\boldsymbol{w}_y^\top\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{x})+b_y+\rho\log\pi_y\right)}{\sum_{j=1}^K\exp\left(\boldsymbol{w}_j^\top\mathcal{F}_{\boldsymbol{\theta}}(\boldsymbol{x})+b_j+\rho\log\pi_j\right)}\right], \tag{21}$$

  where $\rho\geq 0$ is a hyper-parameter that controls the scale of offset to each of the logits. We use $\rho=1.0$ throughout the experiments.

A.2 Classifier Re-Training Methods

Assume that we have pre-trained parameters $\boldsymbol{\Theta}^*=(\boldsymbol{\theta}^*,\boldsymbol{\phi}^*)$ after the first representation learning stage. From this, the classifier re-training methods aim to find a new classifier parameters $\boldsymbol{\phi}^{**}=(\boldsymbol{w}_k^{**},b_k^{**})_{k=1}^K$, while the feature extractor parameters $\boldsymbol{\theta}^*$ is frozen.

- **Classifier Re-Training (cRT)** is the most straightforward way to re-train the classifier from scratch (Kang et al., 2020). Specifically, it first randomly re-initializes the classifier parameters $\boldsymbol{\phi}$ and re-trains them on the class-balanced training dataset $\mathcal{D}_{\text{CB}}$,

$$\boldsymbol{\phi}^{**}=\arg\min_{\boldsymbol{\phi}}\mathbb{E}_{(\boldsymbol{x},y)\sim p_{\mathcal{D}_{\text{CB}}}}\left[-\log\boldsymbol{p}^{(y)}(\boldsymbol{x};(\boldsymbol{\theta}^*,\boldsymbol{\phi}))\right]. \tag{22}$$

- **Learnable Weight Scaling (LWS)** only re-trains the scale of pre-trained weights $(\boldsymbol{w}_k^*)_{k=1}^K$ while the direction is kept (Kang et al., 2020). Specifically, it introduces trainable parameter $\tau\in\mathbb{R}$ which controls the intensity of the weight normalization,

$$\boldsymbol{\phi}(\tau)=(\boldsymbol{w}_k^*/\left\|\boldsymbol{w}_k^*\right\|^\tau,b_k^*)_{k=1}^K. \tag{23}$$

  and finds $\boldsymbol{\phi}^{**}=\boldsymbol{\phi}(\tau^*)$ on the class-balanced training dataset $\mathcal{D}_{\text{CB}}$, where

$$\tau^*=\arg\min_{\tau}\mathbb{E}_{(\boldsymbol{x},y)\sim p_{\mathcal{D}_{\text{CB}}}}\left[-\log\boldsymbol{p}^{(y)}(\boldsymbol{x};(\boldsymbol{\theta}^*,\boldsymbol{\phi}(\tau)))\right]. \tag{24}$$

- **Distribution Alignment (DisAlign)** trains additional modules while the original classifier parameters $\boldsymbol{\phi}^*$ is kept (Zhang et al., 2021). Specifically, it introduces trainable parameters $\{\alpha_k \in \mathbb{R}, \beta_k \in \mathbb{R}\}_{k=1}^K \cup \{\boldsymbol{\gamma} \in \mathbb{R}^K, \delta \in \mathbb{R}\}$ which performs a distribution calibration,

$$z_k(\boldsymbol{x}) \leftarrow \boldsymbol{w}_k^{*\top} \mathcal{F}_{\boldsymbol{\theta}^*}(\boldsymbol{x}) + b_k^*, \tag{25}$$

$$\sigma(\boldsymbol{x}) \leftarrow \mathrm{Sigmoid}(\boldsymbol{\gamma}^\top z_k(\boldsymbol{x}) + \delta), \tag{26}$$

$$\hat{z}_k(\boldsymbol{x}) \leftarrow \sigma(\boldsymbol{x}) \cdot (\alpha_k z_k(\boldsymbol{x}) + \beta_k) + (1 - \sigma(\boldsymbol{x})) \cdot z_k(\boldsymbol{x}), \quad \text{for } k = 1, ..., K, \tag{27}$$

  where z_k and $\hat{z}_k$ respectively denote the original logits and the calibrated logits. While both $\boldsymbol{\theta}^*$ and $\boldsymbol{\phi}^*$ are fixed, it finds the optimal value of $\{\alpha_k, \beta_k\}_{k=1}^K \cup \{\boldsymbol{\gamma}, \delta\} = \{\alpha_k^*, \beta_k^*\}_{k=1}^K \cup \{\boldsymbol{\gamma}^*, \delta^*\}$ on the training dataset $\mathcal{D}$ with Generalized Re-Weighting (GRW).

A.3 Evaluation metrics

The problem we addressed in this paper is the K-way classification problem. Let $\mathcal{P} : \boldsymbol{x} \mapsto \boldsymbol{p}$ be a model that outputs a categorical probability $\boldsymbol{p} \in [0, 1]^K$ for a given input $\boldsymbol{x}$. Following metrics are reported for all of the methods using our implementation.

- **Accuracy (ACC; higher is better):**

$$\mathrm{ACC}(\mathcal{P}, \mathcal{D}) = \mathbb{E}_{(\boldsymbol{x},y)\in\mathcal{D}} \left[\left[y = \arg\max_k \mathcal{P}^{(k)}(\boldsymbol{x}) \right] \right], \tag{28}$$

  where inner $[\cdot]$ denotes the Iverson bracket.
- **Negative log-likelihood (NLL; lower is better):**

$$\mathrm{NLL}(\mathcal{P}, \mathcal{D}) = \mathbb{E}_{(\boldsymbol{x},y)\in\mathcal{D}} \left[-\log \mathcal{P}^{(y)}(\boldsymbol{x}) \right], \tag{29}$$

  which is equivalent to the softmax cross-entropy loss used in training.
- **Expected calibration error (ECE; lower is better):**

$$\mathrm{ECE}(\mathcal{P}, \mathcal{D}; N) = \sum_{n=1}^{N} \delta_n \left(|\mathcal{B}_n| / |\mathcal{D}| \right), \tag{30}$$

  where $\{\mathcal{B}_1, ..., \mathcal{B}_N\}$ is a partition of $\mathcal{D}$,

$$\mathcal{B}_n = \left\{ (\boldsymbol{x}, y) \in \mathcal{D} \;\middle|\; \max_k \mathcal{P}^{(k)}(\boldsymbol{x}) \in \left(\frac{n-1}{N}, \frac{n}{N} \right] \right\}, \quad \text{for } n = 1, ..., N, \tag{31}$$

  and δ_n denotes a calibration error for the nth bin $\mathcal{B}_n$,

$$\delta_n = \left| \mathrm{ACC}(\mathcal{P}, \mathcal{B}_n) - \mathbb{E}_{(\boldsymbol{x},\cdot)\in\mathcal{B}_n} \left[\max_k \mathcal{P}^{(k)}(\boldsymbol{x}) \right] \right|, \quad \text{for } n = 1, ..., N. \tag{32}$$

  We used $N = 15$ bins in this paper.

A.4 Detailed implementation of the proposed method

Loss function During the optimization, we simply fix $M = 10$ and use equal weights for two terms in the final objective, i.e., the actual implementation is

$$\underbrace{0.5\hat{\mathcal{L}}_{\mathrm{CE}}}_{\substack{\text{cross-entropy}\\ \text{term}}} + \underbrace{0.5\hat{\mathcal{L}}_{\mathrm{KD}}}_{\substack{\text{self-distillation}\\ \text{term}}}. \tag{33}$$

Also, we blocked the gradient flow through the target Dirichlet distribution in the self-distillation term, i.e., `jax.lax.stop_gradient(`$\boldsymbol{\beta}$`)` in JAX library. Our training diverges when using the typical softmax outputs due to the sharp target Dirichlet distribution. Thus, we use the following *temperature-scaled* softmax outputs to stabilize the optimization on the self-distillation term $\hat{\mathcal{L}}_{\mathrm{KD}}$,

$$\boldsymbol{p}_{\boldsymbol{\theta}}^{(k)}(\boldsymbol{x}) \leftarrow \frac{\exp\left((\boldsymbol{w}_k^\top \mathcal{F}_{\boldsymbol{\theta}}^{(k)}(\boldsymbol{x}) + b_k) / \tau_{\mathrm{KD}} \right)}{\sum_{j=1}^K \exp\left((\boldsymbol{w}_j^\top \mathcal{F}_{\boldsymbol{\theta}}^{(j)}(\boldsymbol{x}) + b_j) / \tau_{\mathrm{KD}} \right)}, \quad \text{for } k = 1, ..., K. \tag{34}$$

Table 5 shows results for the final version of our algorithm (i.e., SRepr in Section 6.2) on the validation split of ImageNet-LT swept over $\tau_{\mathrm{KD}} \in \{1, 2, 5, 10, 20\}$. Training losses were unstable for $\tau \in \{1, 2, 5\}$ and thus we fix $\tau_{\mathrm{KD}} = 20$ throughout all experiments.

Table 5: Validation results of SRepr with varying temperatures (i.e., τ_{KD} in Appendix A.4) on ImageNet-LT. 'N/A' denotes the training diverges.

	ACC (↑)					
ImageNet-LT (Val)	Many	Medium	Few	All	NLL (↓)	ECE (↓)
$\tau_{\text{KD}} = 1.0$	N/A	N/A	N/A	N/A	N/A	N/A
$\tau_{\text{KD}} = 2.0$	N/A	N/A	N/A	N/A	N/A	N/A
$\tau_{\text{KD}} = 5.0$	66.86±0.10	41.58±0.09	7.48±0.09	46.67±0.05	4.238±0.009	0.294±0.001
$\tau_{\text{KD}} = 10.0$	64.21±0.22	50.51±0.14	32.70±0.38	53.36±0.09	2.032±0.001	0.034±0.000
$\tau_{\text{KD}} = 20.0$	64.20±0.18	50.52±0.14	32.89±0.49	53.40±0.18	2.059±0.049	0.030±0.005

Table 6: Ablation study of proposed methods with various balancing strategies on ImageNet-LT: classification accuracy (ACC), negative log-likelihood (NLL), and expected calibration error (ECE).

	Proposed Methods					
Balancing Strategy	**Repr. learning with SWA** (Section 3)	**Classifier learning with stochastic repr.** (Section 4.1)	**Classifier learning with self-distillation** (Section 4.2)	ACC (↑)	NLL (↓)	ECE (↓)
Without re-training classifier				46.91±0.22	**2.546**±0.009	**0.158**±0.003
	✓			**47.08**±0.12	2.631±0.009	0.187±0.002
CBS (Kang et al., 2020)				50.25±0.18	2.364±0.008	0.110±0.001
	✓			50.95±0.12	2.353±0.012	0.120±0.002
	✓	✓		51.33±0.17	2.340±0.012	0.125±0.003
	✓	✓	✓	**51.66**±0.13	**2.203**±0.009	**0.074**±0.002
GRW (Zhang et al., 2021)				50.77±0.13	2.243±0.007	0.026±0.001
	✓			51.33±0.16	2.220±0.010	0.041±0.002
	✓	✓		51.73±0.11	2.206±0.008	0.056±0.003
	✓	✓	✓	**52.08**±0.04	**2.133**±0.005	**0.019**±0.001
LA (Menon et al., 2021)				50.97±0.13	2.231±0.004	0.063±0.001
	✓			51.62±0.05	2.206±0.009	0.077±0.002
	✓	✓		51.84±0.18	2.208±0.009	0.090±0.003
	✓	✓	✓	**52.12**±0.06	**2.130**±0.006	**0.037**±0.001

Pseudo-code for the proposed method Algorithm 1 summarizes the proposed method in pseudo-code. Note that training SWA-Gaussian with a diagonal covariance has virtually no additional cost over conventional training with SGD. It only requires extra space for storing the first and second moments of the backbone parameters (i.e., lines 1-10). Instead, an additional training cost of our approach compared with the existing methods (Kang et al., 2020; Zhang et al., 2021) come from multiple forward passes of the backbone network during the classifier re-training stage (i.e., lines 11-17). Please refer to Appendix B.5 for further investigation regarding this issue.

B Additional Experiments

B.1 Full Table of Results

Ablation studies of proposed methods (Section 6.1). Table 6 is an extended version of Table 2. It provides the results when we apply the following balancing strategies; CBS, GRW, and LA. The arguments we discussed in Section 6.1 consistently hold for all balancing strategies we considered.

Results on image classification tasks (Section 6.2). Table 7 is an extended version of Table 3. Here, we also provide detailed classification accuracy on three splits introduced in (Liu et al., 2019): Many (a set of classes each with over 100 training examples), Medium (a set of classes each with 20-100 training examples), and Few (a set of classes each with under 20 training examples).

B.2 Additional Results on CIFAR10/100-LT

We also provide the experimental results on CIFAR10-LT and CIFAR100-LT. Table 8 shows that our approach outperforms the baselines in terms of every metric we measured. It clearly demonstrates that the proposed approach consistently benefits from training robust decision boundaries regardless of the scale of datasets.

Table 7: Full restuls on ImageNet-LT and iNaturalist-2018: classification accuracy (ACC), negative log-likelihood (NLL), and expected calibration error (ECE).

	ACC (↑)					
ImageNet-LT	Many	Medium	Few	All	NLL (↓)	ECE (↓)
SGD	66.84±0.26	40.78±0.24	12.05±0.23	46.91±0.22	2.546±0.009	0.158±0.003
+ cRT (Kang et al., 2020)	62.83±0.23	46.92±0.26	26.33±0.16	50.25±0.18	2.364±0.008	0.110±0.001
+ LWS (Kang et al., 2020)	63.23±0.26	47.57±0.24	27.78±0.23	50.91±0.15	**2.197**±0.007	**0.054**±0.001
+ LA (Menon et al., 2021)	60.79±0.20	48.11±0.14	33.20±0.34	50.97±0.13	2.231±0.004	0.063±0.001
+ DisAlign (Zhang et al., 2021)	61.63±0.39	48.68±0.11	32.71±0.45	**51.49**±0.15	2.596±0.012	0.202±0.002
SWA (ours)	67.71±0.11	40.74±0.15	11.01±0.10	47.08±0.12	2.631±0.009	0.187±0.002
+ cRT (Kang et al., 2020)	63.54±0.18	47.68±0.16	26.85±0.28	50.95±0.12	2.353±0.012	0.120±0.002
+ LWS (Kang et al., 2020)	63.51±0.30	48.53±0.07	28.66±0.45	51.60±0.10	2.189±0.007	0.077±0.002
+ LA (Menon et al., 2021)	61.60±0.07	48.70±0.03	33.68±0.34	51.62±0.05	2.206±0.009	0.077±0.002
+ DisAlign (Zhang et al., 2021)	62.43±0.20	49.48±0.15	32.65±0.43	**52.18**±0.11	2.673±0.014	0.215±0.002
+ SRepr (ours)	62.52±0.26	49.44±0.18	32.14±0.41	**52.12**±0.06	**2.130**±0.006	**0.037**±0.001

	ACC (↑)					
iNaturalist-2018	Many	Medium	Few	All	NLL (↓)	ECE (↓)
SGD	76.31±0.52	67.89±0.18	62.24±0.17	66.52±0.05	1.568±0.006	0.048±0.003
+ cRT (Kang et al., 2020)	73.03±0.57	69.09±0.10	66.14±0.23	68.33±0.04	1.537±0.006	0.037±0.002
+ LWS (Kang et al., 2020)	72.35±0.43	70.11±0.34	69.73±0.40	70.19±0.08	**1.386**±0.006	0.030±0.001
+ LA (Menon et al., 2021)	69.90±0.50	69.34±0.14	69.66±0.19	69.49±0.15	1.477±0.005	**0.015**±0.002
+ DisAlign (Zhang et al., 2021)	71.68±0.31	70.73±0.25	69.51±0.39	**70.35**±0.21	1.428±0.006	0.064±0.001
SWA (ours)	77.26±0.25	68.23±0.25	61.87±0.13	66.65±0.10	1.568±0.005	0.071±0.001
+ cRT (Kang et al., 2020)	73.30±0.73	69.22±0.19	66.74±0.25	68.66±0.15	1.546±0.002	0.061±0.002
+ LWS (Kang et al., 2020)	72.82±0.45	70.43±0.25	70.06±0.15	70.53±0.09	1.370±0.002	0.049±0.001
+ LA (Menon et al., 2021)	69.70±0.67	69.47±0.11	69.82±0.68	69.63±0.20	1.466±0.003	0.038±0.001
+ DisAlign (Zhang et al., 2021)	72.34±0.57	71.27±0.10	69.84±0.18	**70.81**±0.10	1.410±0.003	0.076±0.000
+ SRepr (ours)	70.70±0.31	70.83±0.20	70.76±0.32	**70.79**±0.17	**1.353**±0.002	**0.036**±0.002

B.3 Further comparisons with state-of-the-art methods

Increasing the number of training epochs. Throughout the main text, we trained all the competing methods for 100 training epochs on ImageNet-LT, which is sufficient to validate the efficacy of our approach. However, state-of-the-art performances reported in other papers are typically from long training epochs than we set in our experiments. We thus provided the results when the number of training epochs gets in line in Table 4. It clearly demonstrates that ours outperforms baselines by a wide margin. Besides, we also tested our method for ResNeXt-50 architecture (Xie et al., 2017) for a fair comparison with LADE (Hong et al., 2021).

Applying mixup augmentation. Zhong et al. (2021) employed mixup augmentation (Zhang et al., 2018) in decoupled training, which is actually one of the main ingredients to achieving that level of performance they reported. However, the mixup augmentation is not exclusive to their method but applicable to generic approaches, including ours. In Table 4, our approach enhanced by the mixup augmentation indeed outperforms the previous baseline utilizing the mixup (Zhong et al., 2021).

B.4 Combining ours with the existing state-of-the-art methods

Apart from the decoupled learning scheme (Kang et al., 2020) we mainly considered in the main text, there are other groups of methods achieving state-of-the-art performances: a) utilizing multiple experts (Zhou et al., 2020; Xiang et al., 2020; Wang et al., 2021), or b) applying contrastive learning algorithms (Cui et al., 2021; Zhu et al., 2022) for dealing with long-tailed data.

Although our proposed method is based on the decoupled learning scheme, we would like to clarify that we can combine ours with the existing state-of-the-art frameworks due to its simplicity (i.e., it only needs SWAG posterior and classifier re-training). To this end, we tested ours upon existing state-of-the-art code bases by simply 1) obtaining SWAG posterior from the publicly available pre-trained checkpoint with a few SGD iterations (e.g., 10 training epochs) and 2) re-training the classification layer as we proposed in Section 4.2. We adapted the following implementations:

Table 8: Results on CIFAR10-LT and CIFAR100-LT: classification accuracy (ACC), negative log-likelihood (NLL), and expected calibration error (ECE).

	ACC (↑)					
CIFAR10-LT	Many	Medium	Few	All	NLL (↓)	ECE (↓)
SGD	-	-	-	72.02±0.31	1.443±0.103	0.207±0.005
+ cRT (Kang et al., 2020)	-	-	-	80.80±0.03	**0.598**±0.017	**0.041**±0.001
+ LWS (Kang et al., 2020)	-	-	-	80.31±0.07	**0.615**±0.002	0.055±0.002
+ LA (Menon et al., 2021)	-	-	-	77.36±0.14	0.947±0.071	0.146±0.001
+ DisAlign (Zhang et al., 2021)	-	-	-	**81.37**±0.02	**0.600**±0.025	0.054±0.001
SWA (ours)	-	-	-	74.86±0.23	1.201±0.083	0.176±0.007
+ cRT (Kang et al., 2020)	-	-	-	81.72±0.05	0.566±0.006	0.048±0.001
+ LWS (Kang et al., 2020)	-	-	-	80.53±0.11	0.603±0.005	0.056±0.002
+ LA (Menon et al., 2021)	-	-	-	81.78±0.07	0.733±0.008	0.109±0.000
+ DisAlign (Zhang et al., 2021)	-	-	-	81.96±0.05	0.574±0.011	0.052±0.000
+ SRepr (ours)	-	-	-	**82.06**±0.01	**0.542**±0.010	**0.021**±0.001

	ACC (↑)					
CIFAR100-LT	Many	Medium	Few	All	NLL (↓)	ECE (↓)
SGD	68.66±0.38	38.94±0.29	9.50±0.57	40.51±0.32	3.202±0.080	0.333±0.010
+ cRT (Kang et al., 2020)	66.05±0.05	42.14±0.15	15.76±0.22	42.59±0.13	3.271±0.010	0.317±0.002
+ LWS (Kang et al., 2020)	66.24±0.05	41.49±0.15	16.95±0.16	42.79±0.11	3.971±0.004	0.377±0.001
+ LA (Menon et al., 2021)	60.09±0.08	43.34±0.17	27.27±0.19	**44.37**±0.15	**2.636**±0.003	**0.231**±0.001
+ DisAlign (Zhang et al., 2021)	66.90±0.12	42.51±0.12	16.94±0.19	43.37±0.11	4.685±0.004	0.413±0.003
SWA (ours)	72.36±0.19	41.51±0.21	7.97±0.71	42.34±0.18	2.924±0.030	0.301±0.007
+ cRT (Kang et al., 2020)	67.37±0.20	46.81±0.35	20.11±0.33	46.00±0.09	2.953±0.004	0.282±0.000
+ LWS (Kang et al., 2020)	67.09±0.07	48.10±0.07	23.33±0.15	47.31±0.04	3.404±0.003	0.327±0.001
+ LA (Menon et al., 2021)	62.05±0.11	47.11±0.13	31.43±0.24	47.63±0.07	2.224±0.020	0.178±0.002
+ DisAlign (Zhang et al., 2021)	67.68±0.22	48.36±0.02	23.17±0.40	47.56±0.05	3.889±0.000	0.358±0.000
+ SRepr (ours)	66.69±0.01	49.91±0.01	23.31±0.11	**47.81**±0.02	**2.148**±0.009	**0.149**±0.002

Table 9: Results for combining ours with the existing state-of-the-art methods on CIFAR100-LT and ImageNet-LT: classification accuracy (ACC), negative log-likelihood (NLL), and expected calibration error (ECE).

	CIFAR100-LT			ImageNet-LT		
Method	ACC	NLL	ECE	ACC	NLL	ECE
RIDE (Wang et al., 2021)	48.3	2.183	0.168	53.9	2.487	**0.196**
+ SWA + SRepr (ours)	**49.2** (+0.9)	**2.176** (-0.007)	**0.159** (-0.009)	**54.8** (+0.9)	**2.471** (-0.016)	0.198 (+0.002)
RIDE + CMO (Park et al., 2022)	48.5	2.161	0.155	54.1	2.425	**0.187**
+ SWA + SRepr (ours)	**49.1** (+0.6)	**2.051** (-0.110)	**0.118** (-0.037)	**55.0** (+0.9)	**2.421** (-0.004)	0.189 (+0.002)
SADE (Zhang et al., 2022)	49.9	2.015	0.120	59.0	1.826	0.060
+ SWA + SRepr (ours)	**50.1** (+0.2)	**2.008** (-0.007)	**0.117** (-0.003)	**59.2** (+0.2)	**1.824** (-0.002)	**0.059** (-0.001)
BCL (Zhu et al., 2022)	52.0	1.941	0.192	57.2	1.871	0.062
+ SWA + SRepr (ours)	**52.4** (+0.4)	**1.899** (-0.042)	**0.102** (-0.090)	**57.5** (+0.3)	**1.857** (-0.014)	**0.036** (-0.026)

- https://github.com/frank-xwang/RIDE-LongTailRecognition
- https://github.com/Vanint/SADE-AgnosticLT
- https://github.com/FlamieZhu/Balanced-Contrastive-Learning

Table 9 demonstrates that ours indeed improves the existing state-of-the-art methods. A marginal improvement upon multi-expert models (i.e., RIDE, RIDE + CMO, and SADE) is probably due to the ensemble model already providing well-calibrated predictions (Lakshminarayanan et al., 2017; Ovadia et al., 2019). Even if SWA-Gaussian well captures a single modality, naive ensembling significantly benefits from the multi-modal Bayesian marginalization (Wilson and Izmailov, 2020). Nevertheless, we believe the clear improvements upon the contrastive learning approach (i.e., BCL), which uses only a single model, bear out the value of the proposed method.

Table 10: Training costs compared with vanilla decoupled training.

Method	Epoch	Wall-clock time	ACC	NLL	ECE
SWA + cRT	20	647.2 sec	52.28	2.309	0.132
SWA + SRepr (ours)	10	**580.9 sec**	**53.76**	**2.072**	**0.059**
SWA + cRT	40	1294.4 sec	52.47	2.284	0.124
SWA + SRepr (ours)	20	**1161.8 sec**	**53.81**	**2.069**	**0.052**

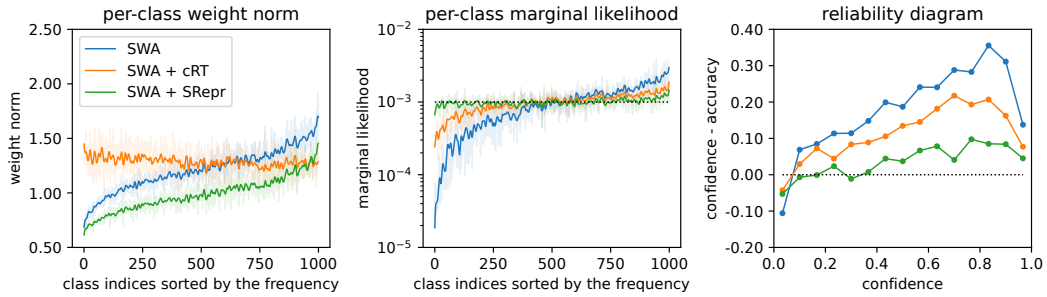

Figure 5: The per-class weight norm of the classifier (left), the per-class marginal likelihood of the test predictions (middle), and the reliability diagram for the test predictions (right). The results are with ResNet-50 on ImageNet-LT.

B.5 Further analysis on proposed methods

Training costs compared with vanilla decoupled training. Our approach (i.e., SRepr) requires M forward passes of the backbone network during the classifier re-training stage. However, the additional training cost due to these multiple forward passes is not a huge bottleneck since we only backpropagate through the last classifier layer while holding the backbone network frozen. To be concrete, Table 10 presents the wall-clock time of re-training classifier with and without SRepr for ResNet-50 on ImageNet-LT when M is 10. While SRepr requires x1.8 times compared to the vanilla classifier re-training (i.e., 32.36 sec/epoch vs. 58.09 sec/epoch), the performance gain achieved by SRepr outweighs the additional training time. That is, even if we run cRT for twice more training epochs than SRepr, it is worse than the performance that SRepr could achieve with the half number of training epochs. We also note that cRT cannot reach the numbers obtained by SRepr even if it is ran for longer training epochs.

Per-class weight norms and marginal likelihoods after classifier re-training Following previous works (Kang et al., 2020; Ren et al., 2020; Alshammari et al., 2022), we further investigate how our proposed approach affects 1) the per-class weight norm, i.e., $\|\boldsymbol{w}_y\|$ for $y = 1, ..., K$, and 2) the per-class marginal likelihood, i.e., $p(y) = \mathbb{E}_{(\boldsymbol{x},\cdot)\sim\mathcal{D}_{\text{test}}}[\boldsymbol{p}^{(y)}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})]$ for $y = 1, ..., K$, where $\mathcal{D}_{\text{test}}$ is a balanced test split. Fig. 5 depicts the per-class weight norm (left) and the per-class marginal likelihood (middle), along with class indices in x-axes sorted by the number of training examples for each class. We also plot the reliability diagram (right; Guo et al., 2017), showing whether the classification model produces well-calibrated predictions.

Alshammari et al. (2022) argued that balanced weight norms of the classifier give tail classes a chance to compete with head classes and produce the ideal marginal likelihood following a uniform distribution. However, Fig. 5 demonstrates that *our re-training method (i.e., SRepr) achieves both the uniform marginal likelihood (middle) and the well-calibrated prediction (right), even though it does not balance weight norms (left)*. This phenomenon suggests that our proposed approach works distinctly from the existing works balancing the weight norms for dealing with long-tailed data.

Measuring dispersion along with class indices While we already confirmed a positive correlation between *per-instance* NLL and dispersion in Section 3.2, it would be worth further investigating it in the context of long-tailed recognition. To this end, Fig. 6 depicts the *per-class* dispersion along with class indices sorted by the number of training examples for each class. It clearly shows that the

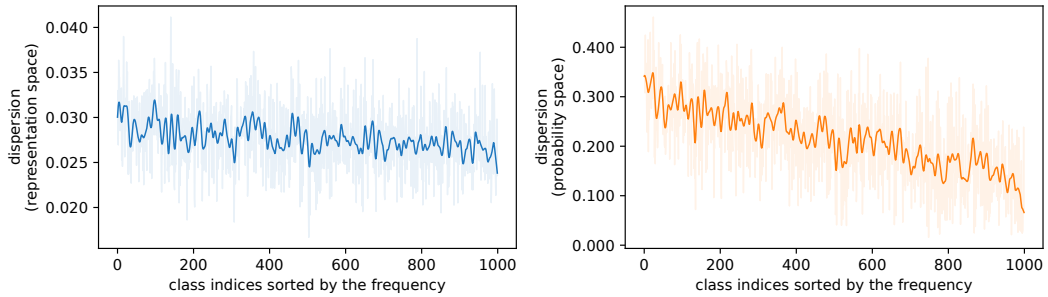

Figure 6: The per-class dispersion along with class indices in the representation space (left) and the probability space (right). The results are with ResNet-50 on ImageNet-LT.

Table 11: Ablation study on mixup augmentation and our approach.

mixup	**SRepr (ours)**	ACC	NLL	ECE
		52.28	2.309	0.132
✓		53.64 (+1.36)	2.131 (-0.178)	0.079 (-0.053)
	✓	53.81 (+1.53)	2.069 (-0.240)	0.052 (-0.080)
✓	✓	**54.43** (+2.15)	**1.990** (-0.319)	**0.022** (-0.110)

head class tends to have smaller dispersion (especially in the probability space dispersion), which indicates that our method is suitable for long-tailed recognition.

Ablation study on mixup augmentation and our approach. We also present the ablation study on the mixup augmentation and our approach (i.e., SRepr) since both methods improve the calibration of the classification model. Table 11 clarifies our effectiveness distinct from the mixup augmentation for ResNet-50 on ImageNet-LT; (1) The performance gain is most significant when we use them together. (2) While both the mixup and SRepr improve all metrics upon the baseline, the contribution from SRepr is more significant than that from the mixup.

Generating stochastic representation by random augmentation. We can also generate stochastic representations by changing inputs (i.e., random augmentation) instead of model parameters (i.e., SWA-Gaussian, as we introduced in Eq. (9)). Table 12 shows that the proposed SRepr also works with random augmentation (i.e., random crop augmentation), but it is worse than one we originally proposed. One notable advantage of generating stochastic representations using SWAG is that we do not need to design data-dependent augmentation strategies. This helps when we are to apply our method to the domain for which no straightforward data augmentation strategies are available (e.g., text, graph, and speech data).

C Experimental Details

Code is available at (TBD; for the moment, we attach it as a separate file). Our implementations are built on JAX (Bradbury et al., 2018), Flax (Heek et al., 2020), and Optax (Hessel et al., 2020). These libraries are available under the Apache-2.0 license[2]. We conduct all experiments on 8 TPUv3 cores, supported by TPU Research Cloud[3].

C.1 Datasets

ImageNet-LT. It is available at https://github.com/zhmiao/OpenLongTailRecognition-OLTR (Liu et al., 2019). It consists of 115,846 train examples, 20,000 validation examples and 50,000 test

[2]https://www.apache.org/licenses/LICENSE-2.0
[3]https://sites.research.google/trc/about/

Table 12: Ablation study on alternative way to generate stochastic representation. $\mathcal{F}_m(\boldsymbol{x})$ denotes the mth stochastic representation for a given input $\boldsymbol{x}$.

Method	How to generate the stochastic representation?	ACC	NLL	ECE
SWA + cRT	-	52.28	2.309	0.132
SWA + SRepr (ours)	changing parameters, i.e., $\mathcal{F}_m(\boldsymbol{x}) = \mathcal{F}_{\boldsymbol{\theta}_m}(\boldsymbol{x})$, where $\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M \overset{\text{i.i.d.}}{\sim} q(\boldsymbol{\theta})$.	**53.81**	**2.069**	**0.052**
SWA + SRepr (ours)	changing inputs, i.e., $\mathcal{F}_m(\boldsymbol{x}) = \mathcal{F}_{\boldsymbol{\theta}_{\text{SWA}}}(\boldsymbol{x}_m)$, where $\boldsymbol{x}_m$ is the mth augmented version of $\boldsymbol{x}$.	53.35	2.115	0.068

examples from 1,000 classes. While the train split is imbalanced, with maximally 1,280 images per class and minimally 5 images per class, the validation and test splits are balanced.

We follow the standard data augmentation policy which consists of random resized cropping with an images size of $224 \times 224 \times 3$ and random horizontal flipping. All images are standardized by subtracting the per-channel mean and dividing the result by the per-channel standard deviation. For per-channel mean and standard deviation values, we stay consistent with the values from the full ImageNet-1k dataset (Russakovsky et al., 2015), i.e., mean of $(0.485, 0.456, 0.406)$ and standard deviation of $(0.229, 0.224, 0.225)$ in RGB order.

iNaturalist-2018. It is available at https://github.com/visipedia/inat_comp (Van Horn et al., 2018). It consists of 437,513 train examples, 24,426 validation examples and 149,394 test examples from 8,142 classes. Since the ground-truth labels of the test split are not publicly available, we instead use the balanced validation split as the test split. We apply the same data augmentation for training as that of ImageNet-LT.

CIFAR10/100-LT. It is available at https://github.com/kaidic/LDAM-DRW (Cao et al., 2019). It consists of 10,847 train examples and 10,000 test examples from 10/100 classes when an exponential decay with an imbalance factor of 0.01 is applied. We use a simple data augmentation policy which consists of random padded cropping and random horizontal flipping (He et al., 2016). All images are standardized with the mean of $(0.4914, 0.4822, 0.4465)$ and standard deviation of $(0.2023, 0.1994, 0.2010)$ in RGB order.

C.2 Optimization

ImageNet-LT and iNaturalist-2018. Throughout the main experiments on ImageNet-LT and iNaturalist-2018, we use an SGD optimizer with batch size 256, Nesterov momentum 0.9, and a single-cycle cosine decaying learning rate starting from the base learning rate of 0.1. Unless specified, the optimization for the representation learning stage terminates after 100 training epochs for ImageNet-LT and 200 training epochs for iNaturalist-2018. For the classifier re-training, we introduce an additional 10% training epochs to re-train the classifier.

CIFAR10/100-LT. Throughout the additional experiments on CIFAR10/100-LT in Appendix B.2, we apply the same optimization strategy as that of ImageNet-LT and iNaturalist-2018, except for the batch size of 128 and the baseline learning rate of 0.5.

C.3 Weight Decay (WD)

Weight Decay (WD; Krogh and Hertz, 1991) is the standard regularization technique for training deep neural networks. For instance, we additionally introduce the WD term of $\lambda_{\text{wd}} \|\boldsymbol{\Theta}\|_2^2$ in Eq. (2), where $\lambda_{\text{wd}} > 0$ is a hyperparameter to control the impact of the WD term. Table 13 shows validation accuracy of SGD swept over $\lambda_{\text{wd}} \in \{0.0001, 0.0002, 0.0003, 0.0004, 0.0005\}$. We empirically found that tuning weight decay exerts a strong influence on long-tailed classification performance, as Alshammari et al. (2022) reported. Throughout the paper, we apply $\lambda_{\text{wd}} = 0.0003$ for ImageNet-LT, $\lambda_{\text{wd}} = 0.0001$ for iNaturalist-2018, and $\lambda_{\text{wd}} = 0.0005$ for CIFAR10/100-LT.

Table 13: Validation accuracy of SGD with varying weight decay coefficients (i.e., λ_{wd} in Appendix C.3) on ImageNet-LT and iNaturalist-2018.

	ImageNet-LT				iNaturalist-2018			
	Many	Medium	Few	All	Many	Medium	Few	All
$\lambda_{\text{wd}} = 0.0001$	64.72	39.27	14.66	45.73	76.31$\pm$0.52	67.89$\pm$0.18	62.24$\pm$0.17	**66.52**$\pm$0.05
$\lambda_{\text{wd}} = 0.0002$	67.99	41.38	13.93	47.90	77.48$\pm$0.39	68.53$\pm$0.14	61.07$\pm$0.38	66.50$\pm$0.17
$\lambda_{\text{wd}} = 0.0003$	69.08$\pm$0.36	41.66$\pm$0.35	12.67$\pm$0.52	**48.28**$\pm$0.22	77.91	67.37	58.46	64.93
$\lambda_{\text{wd}} = 0.0004$	69.29	41.16	10.50	47.83	77.16	65.54	54.37	62.32
$\lambda_{\text{wd}} = 0.0005$	69.44	40.24	8.93	47.23	77.20	63.70	50.85	60.01

Table 14: Validation accuracy of SWA with varying SWA learning rates (i.e., η_{SWA} in Appendix C.4) on ImageNet-LT and iNaturalist-2018.

	ImageNet-LT				iNaturalist-2018			
	Many	Medium	Few	All	Many	Medium	Few	All
$\eta_{\text{SWA}} = 0.020$	70.16	40.69	10.50	47.94	-	-	-	-
$\eta_{\text{SWA}} = 0.010$	70.02$\pm$0.49	41.63$\pm$0.34	11.55$\pm$0.28	**48.47**$\pm$0.22	77.47	67.69	61.49	66.25
$\eta_{\text{SWA}} = 0.005$	69.62	41.55	12.03	48.35	77.26$\pm$0.25	68.23$\pm$0.25	61.87$\pm$0.13	**66.65**$\pm$0.10
$\eta_{\text{SWA}} = 0.001$	-	-	-	-	76.60	67.86	61.71	66.33

C.4 Stochastic Weight Averaging (SWA)

Stochastic Weight Averaging (SWA; Izmailov et al., 2018) has three hyper-parameters; 1) when does the averaging phase start? 2) how frequently update the moving average? and 3) how to set the learning rate during the averaging phase? In response, we follow the instruction from (Izmailov et al., 2018); 1) the averaging phase starts at 75% of the training epoch, 2) we capture parameters at each epoch for averaging, and 3) we use the high constant learning rate η_{SWA} during the averaging phase. Table 14 shows validation accuracy of SWA swept over $\eta_{\text{SWA}} \in \{0.001, 0.005, 0.010, 0.020\}$ on ImageNet-LT. Throughout the paper, we use $\eta_{\text{SWA}} = 0.010$ for ImageNet-LT, $\eta_{\text{SWA}} = 0.005$ for iNaturalist-2018, and $\eta_{\text{SWA}} = 0.1$ for CIFAR10/100-LT.

Algorithm 1 Decoupled training w/ SWA + SRepr (ours).

Require: Train dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, the feature extractor $\mathcal{F}$ parameterized by $\boldsymbol{\theta}$, and the linear classification layer parameterized by $\boldsymbol{\phi} = (\boldsymbol{w}_k, b_k)_{k=1}^K$.
Require: The number of representation learning steps T_1, the number of classifier retraining steps T_2, the learning rate schedule η_t, moment update start time T_{SWA} and update frequency f_{SWA}.
Ensure: Parameters $(\boldsymbol{\theta}^*, \boldsymbol{\phi}^{**})$, trained in decoupled learning scheme.

// Representation learning stage
Initialize the parameters $\boldsymbol{\Theta}_0 = (\boldsymbol{\theta}_0, \boldsymbol{\phi}_0)$ to random values.
Initialize the first and second moments $\boldsymbol{\Theta}_{\text{SWA}} = \mathbf{0}$, $\boldsymbol{\Theta}'_{\text{SWA}} = \mathbf{0}$, and $n_{\text{SWA}} = 0$.
for $t \in \{1, ..., T_1\}$ **do**
    Sample mini-batch $\mathcal{B} \subset \mathcal{D}$.
    Update the parameters with stochastic gradients,

$$\boldsymbol{\Theta}_t \leftarrow \boldsymbol{\Theta}_{t-1} - \eta_t \nabla_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\Theta})\big|_{\boldsymbol{\Theta}=\boldsymbol{\Theta}_{t-1}}, \tag{35}$$

where the loss is defined as

$$\mathcal{L}(\boldsymbol{\Theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{B}} \left[-\log \boldsymbol{p}^{(y)}(\boldsymbol{x}; \boldsymbol{\Theta}) \right]. \tag{36}$$

    if $t > T_{\text{SWA}}$ and $\texttt{MOD}(t, f) = 0$ **then**
        Update the first and second moments via moving average,

$$\boldsymbol{\Theta}_{\text{SWA}} \leftarrow (n_{\text{SWA}} \boldsymbol{\Theta}_{\text{SWA}} + \boldsymbol{\Theta}_t)/(n_{\text{SWA}} + 1), \tag{37}$$

$$\boldsymbol{\Theta}'_{\text{SWA}} \leftarrow (n_{\text{SWA}} \boldsymbol{\Theta}'_{\text{SWA}} + \boldsymbol{\Theta}_t^2)/(n_{\text{SWA}} + 1), \tag{38}$$

$$n_{\text{SWA}} \leftarrow n_{\text{SWA}} + 1. \tag{39}$$

    end if
end for

// Classifier retraining stage
We have the pre-trained parameters $\boldsymbol{\Theta}_{T_1}$, where $\boldsymbol{\Theta}_{T_1} \leftarrow \boldsymbol{\Theta}_{\text{SWA}}$ if we applied SWA.
We also have the approximate posterior over the feature extractor parameters, i.e.,

$$q(\boldsymbol{\theta}|\mathcal{D}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}_{\text{SWA}}, \boldsymbol{\Sigma}_{\text{SWAG}}), \text{ where } \boldsymbol{\Sigma}_{\text{SWAG}} = \text{diag}(\boldsymbol{\theta}' - \boldsymbol{\theta}_{\text{SWA}}^2). \tag{40}$$

for $t \in \{T_1 + 1, ..., T_1 + T_2\}$ **do**
    Sample mini-batch $\mathcal{B} \subset \mathcal{D}$.
    Update the parameters with stochastic gradients (with some balancing strategy),

$$\boldsymbol{\phi}_t \leftarrow \boldsymbol{\phi}_{t-1} - \eta_t \nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi})\big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_{t-1}}, \tag{41}$$

where the loss is defined as

$$\mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{B}} \left[-\log \boldsymbol{p}^{(y)}(\boldsymbol{x}; (\boldsymbol{\theta}^*, \boldsymbol{\phi})) \right], \tag{42}$$

$$\text{or } \mathcal{L}(\boldsymbol{\phi}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{B}} \left[0.5\hat{\mathcal{L}}_{\text{CE}} + 0.5\hat{\mathcal{L}}_{\text{KD}} \right], \text{ if we applied SRepr.} \tag{43}$$

Here, the first term is

$$\hat{\mathcal{L}}_{\text{CE}} = \mathbb{E}_{(\boldsymbol{x},y)\sim q(\boldsymbol{\theta}|\mathcal{D})} \left[-\log \boldsymbol{p}^{(y)}(\boldsymbol{x}; (\boldsymbol{\theta}, \boldsymbol{\phi})) \right]. \tag{44}$$

Refer to Eq. (16) for the detailed definition of the second term.
end for

Return $\boldsymbol{\theta}^* = \boldsymbol{\theta}_{T_1}$ and $\boldsymbol{\phi}^{**} = \boldsymbol{\phi}_{T_1+T_2}$, trained in decoupled learning scheme.