A. Implementation Details

In our experiments, we apply the Wide ResNet-28-2 [30] as the backbone for all the baseline methods and train it for 2.5×10^5 iterations with the batch size 64. In the training of re-balancing baseline approaches, we use SGD as the optimizer, and the learning rate, momentum and weight decay are set to 0.1, 0.9, and 5e-4 respectively. Besides, following [2, 20], we decay the learning rate by 0.01 at the time step 80% and 90% of the total iterations. In the training of SSL baselines cRT, FixMatch, ReMixMatch, VAT, Mean-Teacher, MixMatch and DARP, we follow [20] and use Adam [21] as optimizer with learning rate and weight decay of 2e-3 and 5e-4, respectively. Besides, we use an exponential moving average (EMA) of the model's parameters with the decay rate of 0.999. In the testing stage, we test the trained model every 500 iterations and report the average performance of the last 20 testing results following [2, 20]. For other hyper-parameters settings of the above SSL baselines, please refer [20] for more details. For the training of CReST and CReST+, the combination of FixMatch and MixMatch are trained for 2^{16} and 2^{17} steps respectively in each generation. And we follow [42] to use cosine learning rate decay. For more details of the implementation of CReST and CReST+, please refer [42]. The details of our experiment unfold as follows: We have the same network structure and hyperparameters as DARP, over the various data sets and with different imbalanced ratios, we pick up β from $\{0.9, 0.99, 0.999, 0.9999\}$ for optimal performance. In addition, we start performing dynamic re-weighting for unlabeled data from epoch 200. In the early stage of training, the pseudo number of each class is inaccurate and unreliable. In extreme cases, for some few-shot classes, the pseudo number may be small. This may cause a large variance in the estimation of effective number for unlabeled data. Therefore, in the early stage of training, it is not feasible to re-weight the unlabeled data. In Figure 2 we can see that, $\hat{R}^u$ has much smaller fluctuation after 200 epoch than before. On other hand, in the end-stage of the training, the model is about to converge, so, it is also not effective to apply re-weighting to unlabeled data during this stage. In our experiments, we find that, it is better to estimate the pseudo number from 200 epoch if the total training epoch is set to 500. The threshold $\pi = 0.95$ is chosen as with FixMatch. We use the same weak and strong augmentation as Fixmatch. Specifically, weak augmentation is a standard flip-and-shift augmentation strategy. Strong augmentation includes Cutout, CTAugment, and RandAugment, etc.

Algorithm 1: Dynamic Re-weighting Semi-supervised Learning (DRw)

Input: Labeled Mini-batch $\mathcal{X}$ with B^l labeled data and unlabeled Mini-batch $\mathcal{U}$ with B^u unlabeled data, model $f(\cdot;\boldsymbol{\theta})$ with its parameters $\boldsymbol{\theta}$, total numbers of labeled and unlabeled data in the training set $N^{s,l}$ and $N^{s,u}$, weak and strong augmentation methods $\alpha(\cdot)$ and $\mathcal{A}(\cdot)$, cross-entropy loss function $\mathcal{H}(\cdot,\cdot)$, the condition threshold method $\mathbb{1}(\cdot)$ to indicates whether a condition holds, threshold τ, weighted coefficient λ, effective number of class c in labeled data e_c^l

Result: Updated parameters of the model $\hat{\boldsymbol{\theta}}$.

for $\boldsymbol{x}_i^l,\ y_i^l$ *in* $\mathcal{X}$ **do**
  $\tilde{\boldsymbol{x}}_i^l = \alpha(\boldsymbol{x}_i^l)$; $\tilde{\boldsymbol{y}}_i^l = f(\tilde{\boldsymbol{x}}_i^l;\boldsymbol{\theta})$;
end
for $\boldsymbol{x}_j^u$ *in* $\mathcal{U}$ **do**
  $\tilde{\boldsymbol{x}}_j^u = \alpha(\boldsymbol{x}_j^u)$; $\bar{\boldsymbol{x}}_j^u = \mathcal{A}(\boldsymbol{x}_j^u)$;
  $\tilde{\boldsymbol{y}}_j^u = f(\tilde{\boldsymbol{x}}_j^u;\boldsymbol{\theta})$; $\hat{y}_j^u = \max(\tilde{\boldsymbol{y}}_j^u)$;
end
$\hat{N}_c^{s,u} = \sum_{j=1}^{B^u} \mathbb{1}(c = \arg\max \tilde{\boldsymbol{y}}_j^u)$;
$\hat{N}^{s,u} = \sum_{c=1}^{C} \hat{N}_c^{s,u}$;
$e_c^l = (1-\beta^{\hat{N}_c^{s,l}})/(1-\beta), e_c^u = (1-\beta^{\hat{N}_c^{s,u}})/(1-\beta)$ *having* $\beta \in [0,1)$;
$\mathcal{L}_s = \frac{1}{B^l} \sum_{i=1}^{B^l} \frac{1}{e_c^l} \mathbb{1}(y_i^l = c)\mathcal{H}(y_i^l, \tilde{\boldsymbol{y}}_i^l)$;
$\mathcal{L}_u = \frac{1}{B^u} \sum_{j=1}^{B^u} \frac{1}{e_c^u} \mathbb{1}(\hat{y}_j^u \geq \tau)\mathcal{H}(\hat{y}_j^u, f(\bar{\boldsymbol{x}}_j^u;\boldsymbol{\theta}))$;
return $\hat{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}}(\mathcal{L}_s + \lambda\mathcal{L}_u)$

Table 5: Exploration the effect of adding reweighting at different epochs. We default to adding at the 200th epoch.

Variants	CIFAR-10 (R^l=R^u)			CIFAR-100 (R^l=R^u)	
	R^l=50	R^l=100	R^l=150	R^l=10	R^l=20
DRw + 10th epoch	80.3	74.7	69.8	60.6	55.6
DRw	**84.7**	**79.3**	**74.1**	**61.8**	**57.1**

B. Algorithm Workflow

Our algorithm flowchart is displayed in 1.

C. Ablation Study of Adding Reweighting Earlier

To explore the effect of adding reweighting earlier, we used the dynamic weighting method at the 10th epoch and the final results are shown in Table 5. The results show that adding dynamic weights at 200th epoch is more effective. In the early stage of training, the pseudo number of each class is inaccurate and unreliable. In extreme cases, for some few-shot classes, the pseudo number may be very small. This may cause a large variance in the estimation of effective number for unlabeled data, it will also cause the early training of the network to be more oscillatory. On other hand, in the end stage of the training, the model is about to converge, so, it is also not effective to apply re-weighting to unlabeled data during this stage. In our experiments, we find that, it is better to estimate the pseudo number from 200th epoch.

D. Theoretical Motivation

A similar class reweighting scheme is proposed in [47], their reweighting method is static and we draw on their theoretical results here. Their Theorm 1 is illustrated below, note that all notations are taken from their paper [47] and are consistent.

Theorem 1 *Let $\ell = \ell_{mar}$ be the multi-class margin loss. Recall that $N_i = \sum_{j=1}^{n} \{y_j = i\}$. With probability at least $1-\delta$, we have the generalization bound*

$$R_Q(f) \leq \sup_{q \in Q} \left\{ \hat{R}_q(f) + \sum_{i=1}^{k} q_i p_i \times \left(4k\mathbb{E}\left[\frac{N_i}{p_i n} \hat{\mathfrak{R}}_{N_i}(\Pi_1(\mathcal{F})) \right] + \sqrt{\frac{\log \frac{k}{\delta}}{2p_i^2 n}} \right) \right\}$$

$$R_Q(f) \leq \sup_{q \in Q} \left\{ \hat{R}_q(f) + \sum_{i=1}^{k} q_i p_i \times \left(4k\mathbb{E}\left[\frac{N_i}{p_i n} \hat{\mathfrak{R}}_{N_i}(\Pi_1(\mathcal{F})) \right] + \sqrt{\frac{\log \frac{k}{\delta}}{2p_i^2 n}} \right) \right\}$$

for every f in $\mathcal{F}$ and the excess risk bound

$$\mathcal{E}_Q(\mathcal{F}) \leq 2 \sup_{q \in Q} \sum_{i=1}^{k} q_i p_i \times \left(8k\mathbb{E}\left[\frac{N_i}{p_i n} \hat{\mathfrak{R}}_{N_i}(\Pi_1(\mathcal{F})) \right] + \sqrt{\frac{\log \frac{k}{\delta}}{2p_i^2 n}} \right).$$

$$\mathcal{E}_Q(\mathcal{F}) \leq 2 \sup_{q \in Q} \sum_{i=1}^{k} q_i p_i \times \left(8k\mathbb{E}\left[\frac{N_i}{p_i n} \hat{\mathfrak{R}}_{N_i}(\Pi_1(\mathcal{F})) \right] + \sqrt{\frac{\log \frac{k}{\delta}}{2p_i^2 n}} \right).$$

Based on this theory, they propose the following weights for each class

$$\alpha_i^{(\kappa,c)} = c \left(\frac{\hat{p_i}^{1/\kappa}}{\sum_{j=1}^{k} \hat{p_j}^{1/\kappa}} \right). \tag{11}$$

where $\hat{p_i}$ is proportional to the number of classes N, κ is the hyper-parameter. They adopt a similar reweighting strategy to ours, but the difference is that their reweighting scheme is static and our reweighting scheme is dynamic.