Difficulty-Net: Learning to Predict Difficulty for Long-Tailed Recognition Supplementary Material

	Imb.	Train	Val / Meta	Test
CIFAR-LT	10–200	49–2 / 490	10	100
ImageNet-LT	256	5 / 1280	10	50
Places-LT	996	5 / 4980	10	100

Table 5: The imbalance ratio (Imb.) and the number of samples per class in each datasets. Since the training splits are imbalanced, we show the number of samples in the least and most frequent classes. Note that exactly the same set of images are used for both the validation sets and meta sets. This indicates that meta-learning based methods including ours do not exploit any extra data.

1. More implementation details

1.1. More details on datasets

Table 5 shows the number of samples in training, validation, and test splits in each dataset. As shown in the table, meta-learning (ML) based methods including ours and [18, 29] reuse validation images for constructing S^{meta}, a dataset to be used for meta learning. Note that (1) our proposed method was compared with other ML based methods in exactly the same conditions, (2) we re-ran the experiments using public codes of previous methods and therefore all the methods compared in this paper are evaluated using exactly the same split as shown above, and (3) all the ML based methods including ours do not use any extra data, and therefore they do not receive any unfair benefit compared to other methods by having S^{meta}. All the hyper-parameters were tuned using the validation sets.

1.2. Hyperparameter settings

For CIFAR100-LT, following [36, 6], we use AutoAugment [4] and Cutout [10]. Following [36], we train ResNet-32 [14] for 12.8K steps with a batch size of 128 and an initial learning rate of 0.1. The learning rate is linearly warmed up to 0.2 over the first 400 steps. It is also decayed by 0.1 after 6.4K and 9.6K steps. For the classifier learning stage in decoupled learning methods, we fix the feature extractor and re-train the classifier for 50 steps using class-balanced sampling following [19]. The learning rate used is 0.1 and is decayed by 0.1 after 30 and 40 epochs. For both the stages, we use a weight decay of $1e{-}4$. CIFAR100-LT experiments are done on a single NVIDIA Tesla V100 GPU.

For ImageNet-LT, we follow [51] and train the models for 180 steps with an initial learning rate of 0.05. The batch size used is 128. We use cosine learning rate decay and weight decay of $5e-4$. In decoupled training, for the second stage we only re-train the classifier for 10 steps using batch size 128 and cosine decayed learning rate with an initial value of 0.05. The models are trained on four NVIDIA Tesla V100 GPUs.

For Places-LT, following [51, 24] we load a ResNet-152 pretrained on ImageNet and then finetune it for 30 steps using an initial learning rate of 0.01 and weight decay of $5e-4$. The learning rate is decayed by 0.1 after 10 and 20 steps. The batch size used is 128. For the classifier learning stage, we retrain the classifier for 20 steps with a batch size of 256 and initial learning rate of 0.1, which is cosine decayed. The training is done on four NVIDIA Tesla V100 GPUs.

For all the datasets, we use SGD optimizer with momentum 0.9. For Difficulty-Net learning, we use ADAM optimizer with a learning rate of 0.001 and a weight decay $1e-4$. All implementations are done on PyTorch.

1.3. Designing the Difficulty-Net

As stated in Sec. 3.2, our Difficulty-Net is a MLP with 2 hidden layers. The illustration of our Difficulty-Net is given in Fig. 5. The output layer dimension changes with the number of classes in the dataset. Here we provide a simple way to select the hidden layer dimensions H. To come up with the method, we compare the end-to-end training performance using different values for H on 2 different datasets. The results are given in Table 6.

We find that the best working H is different for different datasets. Therefore, based on the results, we decide to select $H = 2^n$ such that $2^{n-1} \leq C < 2^n$, where C is the number of classes and n is a positive integer.
The value of C and H for the three different datasets that we used are given in Table 7.

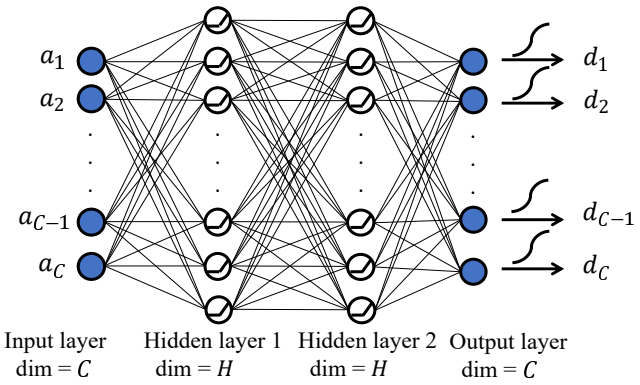

Figure 5: Illustration of our Difficulty-Net

H	CIFAR100-LT ($C = 100$)	ImageNet-LT ($C = 1000$)
128	<u>47.96</u>	39.4
256	**48.06**	40.4
512	47.81	<u>41.2</u>
1024	47.34	**41.4**
2048	46.92	40.8

Table 6: Effect of H on e2e training of ResNet-32 and ResNet-10 on CIFAR100-LT (imbalance=100) and ImageNet-LT respectively.

Dataset	C	H
CIFAR100-LT	100	128
ImageNet-LT	1000	1024
Places-LT	365	512

Table 7: C and H for datasets used in our experiments.

1.4. Algorithm for meta-learning via Difficulty-Net

The algorithm for our Difficulty-Net based learning is provided in Algorithm 1. As stated in Sec. 3.2, our learning method comprises of three main steps (Eq. 9,10 and 11) that are represented by steps 7, 8 and 10 in the algorithm. Note that in our algorithm, S^{meta} is reused as validation set S^{val} for calculating accuracies.

Algorithm 1 Meta-learning using Difficulty-Net

Require: Training set S^{train}, Meta dataset S^{meta}
Require: Initial learnable parameters θ_1 and ϕ_1
Require: Max iterations T, Value of λ
Require: Learning rates α, β and batch sizes b, m
1: **for** $t = 1 \dots T$ **do**
2: Compute $A_{C,t}$ using $f(x; \phi_t)$ on S^{meta}
3: Sample mini-batch of size b from S^{train}
4: Sample mini-batch of size m from S^{meta}
5: Compute weights with $A_{C,t}$ and θ_t using Eq. 4
6: Compute intermediate $\hat{\phi}_t(\theta_t)$ using Eq. 9
7: Update θ_t to θ_{t+1} using Eq. 10
8: Re-compute Eq. 4 with $A_{C,t}$ and θ_{t+1}
9: Update ϕ_t to ϕ_{t+1} using Eq. 11
10: **end for**
Output: ϕ_{T+1}, θ_{T+1}

2. More Results

2.1. CIFAR100-LT results without using extra augmentations

For the CIFAR100-LT results reported in Table 1, we used extra augmentations (AutoAugment [4] and Cutout [10]) to ensure same training setups as recent SOTA methods such as PaCo[6], BALMS [26] and DRO-LT[28] for fair comparison. As expected, these additional augmentation techniques provide a significant boost in the results. To verify that our proposed method is effective independent of these extra augmentations, we compare the results of our method with other SOTA methods without using the augmentation techniques. The results are reported in Table 8. With or without extra augmentations, Ours + LAS proves to be very effective.

2.2. ImageNet-LT results on many-, med- and few-shot classes

In Table 2, we saw that our proposed method helps to achieve the best overall accuracy. Here we study the effectiveness of our method for each of many-, medium- and few-shot classes. The comparison results are given in Table 9. We find that in both e2e learning and decoupled learning, Difficulty-Net based weight assignment helps to significantly boost the performance of the few-shot and medium-shot classes. We believe this result indicates the strong capability of Difficulty-Net based weighting in mitigating biased performance caused by the class imbalance. Especially, Ours + LAS is the most effective for the few-shot classes, irrespective of the model used.

2.3. ImageNet-LT Results Using RandAugment

In Table 2, we reproduced the results of PaCo [6] without using RandAugment [5] for the sake of fair comparison with all the other methods that do not use RandAugment. However, the originally reported results in [6] use RandAugment as additional augmentation, which are significantly higher than the reproduced results. This suggests that PaCo is greatly benefited by the use of RandAugment. Therefore, we used RandAugment with our method and compared the results with PaCo in Table 10. We only used Ours + LAS for

	Imbalance				
Method	200	100	50	20	10
e2e training					
Focal Loss [23] †	35.62	38.41	44.32	51.95	55.78
MWN [29] †	37.91	42.09	46.74	54.37	58.46
Class-Balanced [7] †	36.23	39.60	45.32	52.99	57.99
CB-DA [18] †	39.31	43.35	48.53	55.62	59.58
LDAM [2] †	–	39.60	–	–	56.91
EQL [36] †	37.34	40.54	44.70	54.12	58.32
CDB-CE [31] †	37.40	42.57	46.78	54.22	58.74
PaCo [6]	36.96	40.92	46.97	53.66	59.59
+ Bal. Softmax [26]	39.55	44.13	48.60	55.89	60.24
Ours	39.94	43.82	49.00	55.70	60.25
+ Bal. Softmax	41.43	45.81	51.14	56.58	<u>61.33</u>
decoupled learning					
cRT [19]	40.13	44.04	48.97	55.67	59.54
LWS [19]	40.70	45.05	49.70	56.22	60.00
LAS [51]	40.76	45.32	49.96	56.66	59.96
BALMS [26]	39.58	44.64	48.52	54.28	58.34
MWN + cRT	40.57	44.00	49.47	56.05	59.64
MWN + LWS	40.48	44.52	49.10	55.89	59.48
MWN + LAS	40.94	44.64	49.15	55.91	59.24
Ours + cRT	41.12	45.41	50.50	56.30	60.86
Ours + LWS	<u>41.67</u>	<u>46.04</u>	<u>51.27</u>	<u>56.66</u>	61.30
Ours + LAS	**42.19**	**46.42**	**51.60**	**56.82**	**61.47**

Table 8: Top-1 classification accuracy (%) on CIFAR100-LT without using extra augmentation *i.e*. AutoAugment and Cutout. † denotes copied results from origin paper [31, 18]. The best results are made bold while the second best results are underlined, which applies for the other tables as well.

the comparison because Ours + LAS is the best performing decoupled learning method as seen in Table 1,2 and 3.

From Table 10, we find that using RandAugment benefits our method as well. With or without RandAugment [5], Ours+LAS outperformed PaCo.

2.4. Comparison on ImageNet-LT with MoE methods

In Sec. 4.4, we did not compare our proposed method directly to mixture of experts (MoE) methods as the latter uses multiple experts while we focus on improving the learning of a single expert. For the fair comparison with MoE methods, we created an ensemble of Difficulty-Net based trained models. For the ensemble creation, we trained two expert models using Ours + LAS decoupled learning. The backbone architectures of these two models were kept the same. The only difference between these models was that one used a linear classifier and the other used a cosine classifier. During inference, we simply took the mean outputs of these two models. The results of this simple ensemble is provided in Table 11.

As can be seen, although our ensemble comprises of only two expert models, it performs significantly better than 3-experts and 4-experts RIDE [41]. This shows that our proposed Difficulty-Net is effective in learning expert models for MoE methods. However, the current ensemble is heuristic and a detailed research on contribution of Difficulty-Net in MoE is left for the future.

2.5. More results on sample-level v/s class-level difficulty

As empirically verified in Table 4, class-level difficulty is more effective than sample-level difficulty in our Difficulty-Net. We believe that this happens because as stated in [31] and Sec. 4.5, the absolute number of hard samples in head classes is significantly higher than that in tail classes due to the inherent long-tail characteristic of the dataset. Using sample-level difficulty gives high weights to all the hard samples irrespective of their classes, resulting in more weights for the head classes and therefore getting the model biased to the head classes.

We verified this by conducting a simple experiment on CIFAR100-LT. For 2 classes A and B with 376 and 46 training samples respectively, the absolute number of hard samples given high weights by sample-level method was higher for A(50) than B(13). Although higher proportion of samples in B($\approx$ 28%) received high weights compared to A ($\approx$ 13%), A got more weights compared to B due to its higher absolute number of hard samples. As a result, the accuracy for A is improved from 46% to 62% and that for B is decreased from 31% to 21%, hence boosting the bias. In such case, using class-level difficulty gives high weights to all samples of B, resulting in more weights for B. As a result, the accuracy on B was improved from 31% to 40%, while that on A was almost maintained (46% to 44%).

The effectiveness of class-level difficulty in Difficulty-Net for overcoming model bias is further verified in Table 12. Using sample-level difficulty causes the model to get biased towards the many-shot classes while class-level difficulty is particularly useful for improving performance on the med-shot and few-shot classes.

Backbone Network	ResNet-10				ResNet-50			
Method	Many	Medium	Few	Overall	Many	Medium	Few	Overall
e2e training								
CE	57.6	25.7	3.2	34.8	64.0	38.8	5.8	41.6
Focal Loss [23]	36.4	29.9	16.0	30.5	–	–	–	–
OLTR [24]	43.2	35.1	18.5	35.6	–	–	–	–
EQL [36]	–	–	–	36.4	–	–	–	–
CDB-CE [31]	–	–	–	38.5	–	–	–	–
Bal. Softmax [26]	55.8	35.7	20.9	41.1	–	–	–	–
PaCo[6]*	–	–	–	–	**68.4**	44.8	14.7	49.8
+ Bal. Softmax [26]*	–	–	–	–	59.9	**52.6**	36.1	53.5
Ours	**58.8**	36.4	13.9	41.4	68.1	47.2	21.5	51.2
+ Bal. Softmax	54.6	41.6	27.8	44.3	63.6	51.4	35.8	53.7
decoupled learning								
cRT [19]	–	–	–	41.8	58.8	44.0	26.1	47.3
LWS [19]	–	–	–	41.4	57.1	45.2	29.3	47.7
MiSLAS [51]	–	–	–	–	61.7	51.3	35.8	52.7
BALMS [26]	50.3	39.5	25.3	41.8	–	–	–	–
Ours + cRT	53.3	41.1	27.4	43.6	63.2	51.8	35.2	53.5
Ours + LWS	51.6	**43.7**	29.3	44.4	62.5	52.3	36.6	53.7
Ours + LAS	51.8	43.6	**30.2**	**44.6**	62.9	**52.6**	**36.8**	**54.0**

Table 9: Top-1 accuracy (%) on many-, medium- and few-shot classes of ImageNet-LT. * represents results reproduced using author's codes without using RandAugment [5] for fair comparison. Other results are copied from original papers.

Method	ResNet-10	ResNet-50
PaCo + Bal. Softmax [6]	–	57.0
Ours + LAS	**46.9**	**57.4**

Table 10: Top-1 accuracy (%) using RandAugment[5]. Baseline results are copied from the original paper [6].

Method	ResNet-10	ResNet-50
LFME [45]	38.8	–
RIDE (2 experts) [41]	–	54.4
RIDE (3 experts) [41]	–	54.9
RIDE (4 experts) [41]	–	55.4
Ours (2 experts)	**47.5**	**56.2**

Table 11: Comparison with mixture of expert methods. Baseline results are copied from the original papers [45, 41].

Imbalance	100				10			
Difficulty	Many	Med	Few	All	Many	Med	Few	All
Sample-level	**67.00**	47.46	18.99	45.76	**74.50**	61.01	48.02	62.51
Class-level (ours)	64.47	**51.21**	**24.92**	**47.96**	70.48	**64.40**	**53.36**	**63.52**

Table 12: Comparison of sample-level difficulty and class-level difficulty on CIFAR100-LT.