# A Amortizing the learning of weight vector by introducing a weight net

To integrate our proposed method with deep learning frameworks, we adopt a stochastic setting, $i.e.$, a mini-batch setting at each iteration. We adopt two-stage learning, where stage 1 trains the model $f(\boldsymbol{\theta})$ by the standard cross-entropy loss on the imbalanced training set and stage 2 aims to learn the weight vector $\boldsymbol{w}$ and meanwhile continue to update the model $f(\boldsymbol{\theta})$. Generally, at stage 2, calculating the optimal $\boldsymbol{\theta}$ and $\boldsymbol{w}$ requires two nested loops of optimization, which is cost-expensive. We optimize $\boldsymbol{\theta}$ and $\boldsymbol{w}$ alternatively, corresponding to (1) and (10) respectively, where $\boldsymbol{w}$ is maintained and updated throughout the training, so that re-estimation from scratch can be avoided in each iteration. We summarize the amortized learning of $\boldsymbol{w}$ in Algorithm 2, where the key steps are highlighted in Step (a), (b), and (c). Specifically, at each training iteration $t$, in Step (a), we have $\hat{\boldsymbol{\theta}}^{(t+1)}(\boldsymbol{\Omega}^t) = \{\hat{\boldsymbol{\theta}}_1^{(t+1)}(\boldsymbol{\Omega}^t), \hat{\boldsymbol{\theta}}_2^{(t+1)}(\boldsymbol{\Omega}^t)\}$ and $\alpha$ is the step size for $\boldsymbol{\theta}$; in Step (b), as the cost function based on features is related with $\hat{\boldsymbol{\theta}}_1^{(t+1)}(\boldsymbol{\Omega}^t)$, the OT loss relies on $\hat{\boldsymbol{\theta}}_1^{(t+1)}(\boldsymbol{\Omega}^t)$, and $\beta$ is the step size for $\boldsymbol{\Omega}$; in Step (c), we ameliorate model parameters $\boldsymbol{\theta}^{(t+1)}$.

**Algorithm 2** Workflow about our re-weighting method for optimizing $\boldsymbol{\theta}$ and weight net.

**Require:** Datasets $\mathcal{D}_{\text{train}}$ , $\mathcal{D}_{\text{meta}}$ , initial model parameter $\boldsymbol{\theta}$ and weight net $\boldsymbol{\Omega}$, hyper-parameters $\{\alpha, \beta, \lambda\}$
**for** $t = 1, 2, ..., t_1$ **do**
  Sample a mini-batch $B$ from the training set $\mathcal{D}_{\text{train}}$ ;
  Update $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_B$ where $\mathcal{L}_B = \frac{1}{|B|} \sum_{i \in B} \ell \left( y_i, f\left( x_i; \boldsymbol{\theta}^{(t)} \right) \right)$;
**end for**
**for** $t = t_1 + 1, ..., t_1 + t_2$ **do**
  Sample a mini-batch $B$ from the training set $\mathcal{D}_{\text{train}}$ ;
  Compute weight vector $\boldsymbol{w}^t$ by weight net $\boldsymbol{\Omega}^t$;
  **Step (a):** Update $\hat{\boldsymbol{\theta}}^{(t+1)}(\boldsymbol{w}^{(t)}) \leftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_B$ where $\mathcal{L}_B = \frac{1}{|B|} \sum_{i \in B} w_i^{(t)} \ell \left( y_i, f\left( x_i; \boldsymbol{\theta}^{(t)} \right) \right)$
  Use $\mathcal{D}_{\text{meta}}$ to build $Q$ in (12) and $B$ with $\boldsymbol{w}^t$ to build $P(\boldsymbol{w}^t)$ (4);
  **Step (b):** Compute $L_{\text{OT}} \left( \hat{\boldsymbol{\theta}}_1^{(t+1)}(\boldsymbol{\Omega}^t), \boldsymbol{\Omega}^t \right)$ with cost (9); Optimize $\boldsymbol{\Omega}^{t+1} \leftarrow \boldsymbol{\Omega}^t - \beta \nabla_{\boldsymbol{\Omega}} L_{\text{OT}} \left( \hat{\boldsymbol{\theta}}_1^{(t+1)}(\boldsymbol{\Omega}^t), \boldsymbol{\Omega}^t \right)$
  **Step (c):** Update $\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_B$ where $\mathcal{L}_B = \frac{1}{|B|} \sum_{i \in B} w_i^{(t+1)} \ell \left( y_i, f\left( x_i; \boldsymbol{\theta}^{(t)} \right) \right)$ and we compute $w_i^{(t+1)}$ with updated weight net $\boldsymbol{\Omega}^{t+1}$
**end for**

# B More details and results about the imbalanced image classification

## B.1 Details about imbalanced training datasets

***CIFAR-LT*** is the long-tailed version of CIFAR dataset, where the original CIFAR-10 (CIFAR-100) dataset [38] has 5000 (500) images per class and falls into 10 (100) classes. Following [5], we create long-tailed training sets from CIFAR-10 and CIFAR-100 by discarding some training samples, where we vary the imbalance factor IF $\in \{200, 100, 50, 20\}$. We do not change the balanced test sets and we randomly select ten training images per class as our meta set.

We build ***ImageNet-LT*** based on the classic ImageNet [39] following [24]. After discarding some training examples, ImageNet-LT has 115.8K training examples in 1,000 classes, with an imbalance factor IF $= 1280/5$. The authors also provided a small balanced validation set with 20 images per class, from which we sample 10 images to construct our meta set as [5]. Besides, the original balanced ImageNet validation set is adopted as the test set (50 images per class).

***Places-LT*** is created from Places-2 [40] by [24] with the same strategy as above. The Places-LT contains 62.5K images from 365 categories with $= 4980/5$. This factor means that Places-LT is more changeling than ImageNet-LT dataset. Similarly, there is an official balanced validation dataset with 20 images per class, where we random select 10 images to build our meta set following [4].

***iNaturalist 2018*** [3] is collected from real world which contains 435,713 training samples in 8142 categories with the imbalance factor IF of $1000/2$. Following [4, 5], we use the original validation dataset to evaluate our method and random select 2 samples from every category in training dataset to construct our meta set.

### B.2 Experimental details and results on iNaturalist 2018

Following [4], we use ResNet-50 pre-trained on ImageNet plus iNaturalist 2017 as the backbone network on iNaturalist 2018. For stage 1, we set the initial learning rate as 0.01, which is decay by $1e^{-1}$ every 20 epochs. In the stage 2 of our method, we only fine-tune the last fully connected layer and set $\alpha$ as $1e^{-4}$ and $\beta$ as $1e^{-3}$ within 200 epochs. The mini-batch size is 64, the optimizer is SGD with momentum $0.9$ and weight decay is $5e^{-3}$.

Considering the label vector from the iNaturalist 2018 dataset is high-dimensional and sparse, we only use the features of samples to define the cost function, $i.e.$, *Feature-aware cost*. For training efficiency, we use *Prototype* to construct our meta set. As summarized in Table 6, our proposed method outperforms all the baselines. This observation demonstrates that our proposed re-weighting method can effectively deal with the extremely imbalanced large-scale training set with a large number of classes.

### B.3 Learned weights for different classes with varying imbalanced settings

To explore the learned example weights, we average them within each class and visualize the averaged weight for each class with varying imbalanced factors in Fig. 3. We see that the learned weights of the class 10 (tail class) are more prominent than those of the majority classes. This phenomenon becomes increasingly evident as the training set becomes more imbalanced.

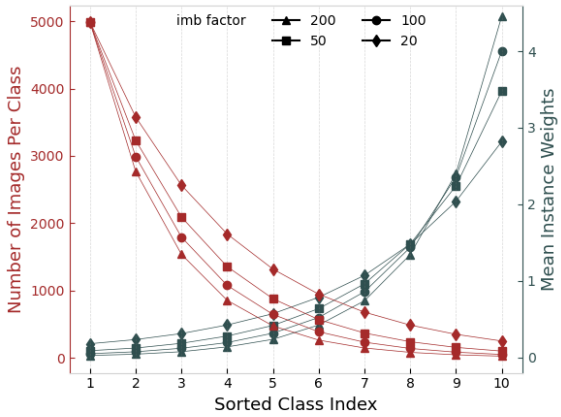


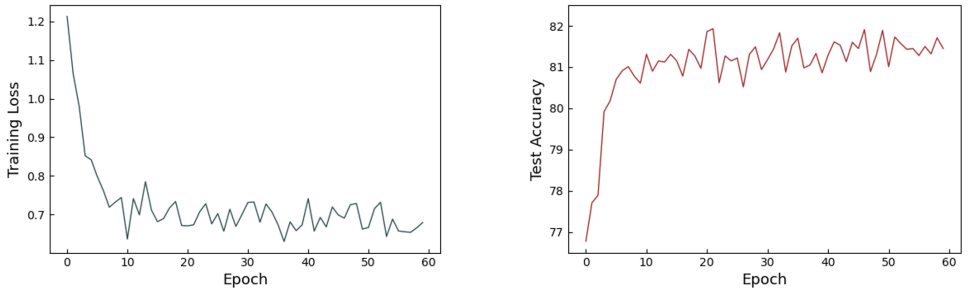

Figure 3: The number of samples and learned mean weights within each class with varying IF on CIFAR-LT-10.

### B.4 Convergence and time complexity

In this section, we investigate the convergence and time complexity. The optimization of the OT distance between discrete distributions of size $n$, is a linear program and can be solved with combinatorial algorithms such as the simplex methods, which dampens the utility of the method when handling large datasets. Recently, the regularization scheme proposed by Cuturi in [37] allows a very fast computation of a transportation plan, denoted as Sinkhorn algorithm. The Sinkhorn algorithm requires the computational cost of $n^2 \log(n)/\varepsilon^2$ to reach $\varepsilon$-accuracy, which is adopted by us to solve the OT problem.

As shown in Fig. 4, we plot the variation of training loss and test classification performance as the increase of training time at the second stage. The training loss of our proposed method can converges at around 10th epoch, where 20 epochs are usually enough for our method to yield satisfactory performance. This means our method can reach fast convergence.

[3] https://github.com/visipedia/inat_comp/tree/master/2018




(a) (b)

Figure 4: Imbalanced training loss (a) and balanced test accuracy (b) on CIFAR-10 with imbalanced factor 100.

Table 6: Test top-1 errors(%) of ResNet-50 on iNaturalist 2018. $*$ indicates results from [5].

| Method | iNaturalist 2018 |
|---|---|
| CE | 34.24 |
| CB, CE*[13] | 33.57 |
| LDAM* [17] | 34.13 |
| LDAM-DRW* [17] | 32.12 |
| cRT* [6] | 32.40 |
| Mets-class-weight, CE*[4] | 32.45 |
| MetaSAug, CE* [5] | 31.25 |
| **Our method** | **30.59** |

To investigate the time complexity, we conduct experiments on CIFAR 10, CIFAR100, ImageNet-LT and Places-LT with different imbalanced factor for ten epoches and average over the time. Models on CIFAR are evaluated with batch size of 16 on a single RTX 3080 GPU. Models on ImageNet-LT are evaluated with batch size of 128 on 4 RTX 2080Ti GPU. Models on Places-LT and iNaturalist 2018 are evaluated with batch size of 32 and 64 on 4 Tesla V100 GPU, respectively. The results are summarized in Table 7. We observe that our proposed method has an acceptable time complexity compared with the cross entropy. It means that our proposed method achieves a better performance with an acceptable cost by introducing the OT loss for re-weighting.

### B.5 Learned Transport plan matrix and weight vector

To explain the effectiveness of learning the weight vector with OT, we give a toy example. The imbalanced training mini-batch consists of $\{10, 9, ..., 1\}$ training samples from class $\{1, 2, ..., 10\}$ and the balanced meta set consists of $\{10, 10, ..., 10\}$ validation samples from class $\{1, 2, ..., 10\}$, which are used to compute the prototype for each class. The process of learning the weight vector is shown in Fig. 5, where we use feature-aware cost, label-aware cost and combined cost in (a)-(c), respectively. For all experiments, we fix the probability measure of meta set as uniform distribution, initialize the to-be-learned probability measure ($i.e.$, $\boldsymbol{w}$) of training mini-batch with a uniform measure, and we then optimize $\boldsymbol{w}$ by minimizing the OT loss with given cost function. Given different cost functions, we find that the learned weight vectors and transport probability matrices show different characteristics. Given the feature-aware cost function, we can assign weight to examples instance-wisely. However, the inaccurate discrimination of features may mislead the learning of weight. Under the label-aware cost function, the learned weights are class-level and inversely related to the class frequency. Given the combined cost function, the results combine the benefits of label-aware and feature-aware cost functions. Interestingly, the weights assigned to examples of tail classes are more prominent than those of the head classes, where the weights of head classes are below the initialization $1/55$ and those of tail classes are above $1/55$. Such results demonstrate our re-weighting method can pay more attention to the tail classes.

Table 7: Running time(s) of our method and the baseline method on CIFAR-LT-10, CIFAR-LT-100, ImageNet-LT, Places-LT and iNaturalist (iNat) 2018 under different settings.

| Datasets | CIFAR-LT-10 | | | | CIFAR-LT-100 | | | | ImageNet-LT | Places-LT | iNat 2018 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Imbalance Factor** | 200 | 100 | 50 | 20 | 200 | 100 | 50 | 20 | 1280/5 | 4980/5 | 1000/2 |
| Cross-entropy (CE) | 11.68 | 13.06 | 14.91 | 18.22 | 9.70 | 11.53 | 13.03 | 16.50 | 241.46 | 375.46 | 3813.22 |
| Our method (Weight Vector) | 13.70 | 15.26 | 17.36 | 21.18 | 11.56 | 13.43 | 15.44 | 19.95 | 252.16 | 381.09 | 4237.73 |

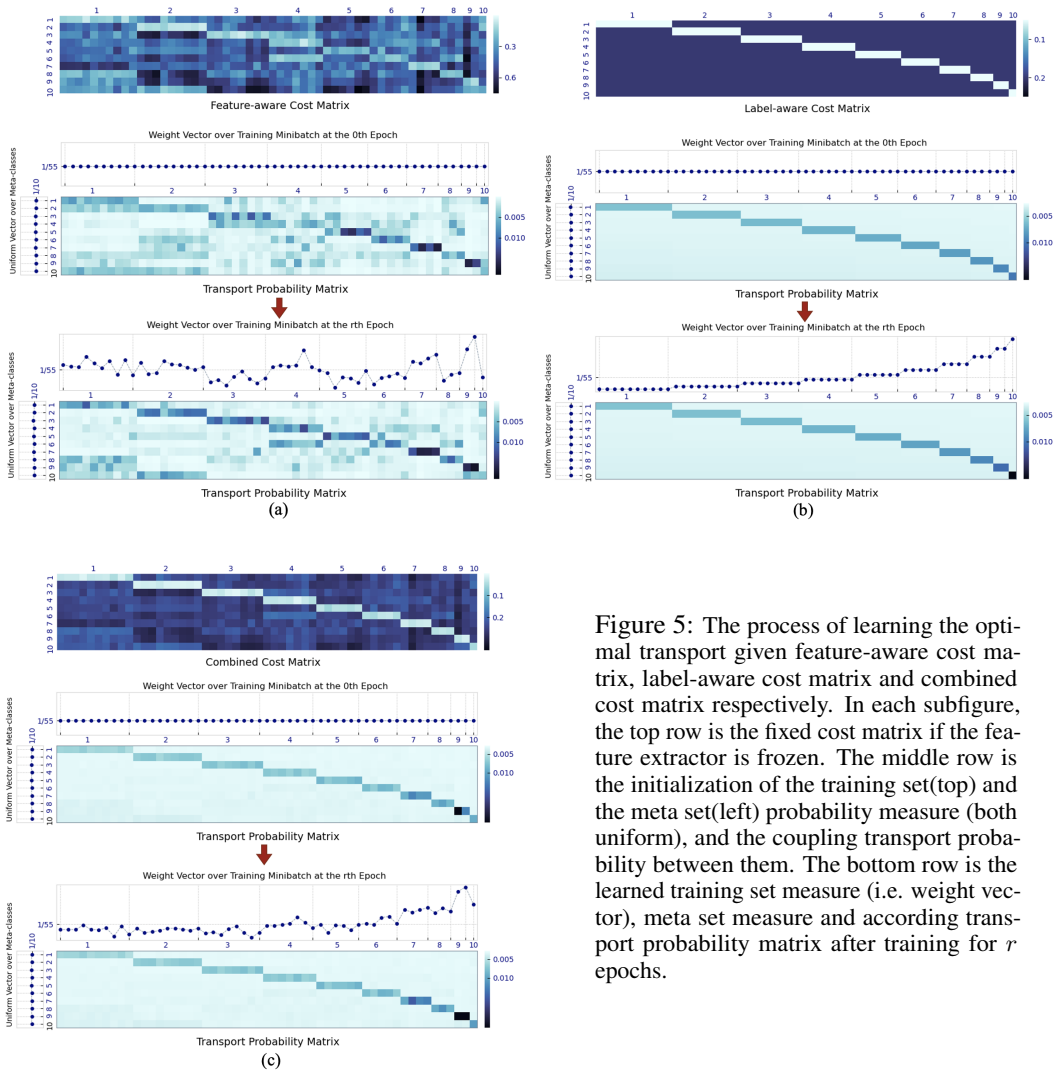



Figure 5: The process of learning the optimal transport given feature-aware cost matrix, label-aware cost matrix and combined cost matrix respectively. In each subfigure, the top row is the fixed cost matrix if the feature extractor is frozen. The middle row is the initialization of the training set(top) and the meta set(left) probability measure (both uniform), and the coupling transport probability between them. The bottom row is the learned training set measure (i.e. weight vector), meta set measure and according transport probability matrix after training for $r$ epochs.

Figure 6: The weight net structure used in text classification experiment.

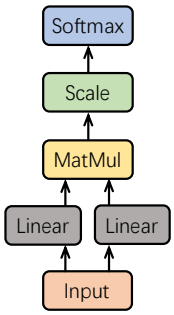



Table 8: Hidden shape in weight net. What we feed into the net is the output vector from feature extractor in backbone network. We give a detailed shape information in weight net when assuming $d_l$ is 128 and $d_f$ is $[32, 768]$, where 32 represents batch size.

| Layer | Shape |
|---|---|
| Input | [32, 768] |
| Linear | [32, 128] |
| MatMul | [32, 32] |
| Scale | [32, 32] |
| Softmax | [32, 32] |

Table 9: The network structure for 3-layer FCN on text classification experiments. **cls_num** is 2 for SST-2 and 5 for SST-5.

| | Input 768-dimensional text features from BERT |
|---|---|
| Feature extractor | Linear layer (768, 768), ReLU |
| Feature extractor | Linear layer (768, 768), ReLU |
| Classifier | Linear layer (768, cls_num), Tanh |

Table 10: Settings of the training on SST-2 and SST-5 dataset.

| Dataset | Fine-tune / Pretrain | Stage 1 | Stage 2 |
|---|---|---|---|
| SST-2 or SST-5 | Adam ($5e^{-6}$)<br>epochs:5<br>batch size:8 | SGD ($1e^{-4}$)<br>epochs:15<br>batch size:50 | SGD ($\alpha$ $1e^{-4}$) and SGD ($\beta$ $4e^{-2}$)<br>epochs:10<br>batch size:50 |

# C More details about the text classification

## C.1 FCN structure

We give the FCN structure in Table 9.

## C.2 Weight net structure on text classification task

We give the weight net structure used in text classification task in Fig. 6. Different that used in image and point cloud classification tasks, we build a self attention-similar weight net and take the sample features $z_i^{\text{train}}$ as input:

$$\boldsymbol{w} = \operatorname{softmax}\left(\boldsymbol{s}/\sqrt{d_l}\right), s^i = \left(\mathbf{W}_1 \boldsymbol{z}_i^{\text{train}}\right)\left(\mathbf{W}_2 \boldsymbol{z}_i^{\text{train}}\right)^T, \tag{13}$$

Specifically, the output of dimension $d_f$ from feature extractor will be encoded to the same dimension $d_l$ by two separate linear layers $\mathbf{W}_1$ and $\mathbf{W}_2$. We compute their dot products, divide by a scaled factor $\sqrt{d_l}$ and apply a softmax operator to obtain expected instance weights. The hidden shape in our experiments is shown in Table 8.

## C.3 Experimental settings

For fining tune BERT, we set the initial learning rate as $5e^{-6}$, where epochs is 5 and batch size is 8. For the first stage, we use SGD optimizer with learning rate $1e^{-4}$. For the second stage, we use SGD optimizer with $\alpha$ learning rate $1e^{-4}$ and $\beta$ learning rate $4e^{-2}$. For both two stages, our minibatch size is 50 and epochs are 15 and 10, respectively. The settings of the training process on the STT-2 and SST-5 are listed in Table 10.

## C.4 Ablation Study

To prove the robustness of our method and its variants on text data, we execute an ablation study on SST-2 dataset, which is shown in Table 11. We can find that combined cost achieves best performance and using label or feature cost can still obtain an acceptable result, which indicates the OT cost is flexible and robustness. When comparing the influence of different ways in constructing meta set, we can see that both whole meta set and prototype-based meta set achieve a close performance, while random sample meta cause a noticeable performance drop.

# D More details about the point cloud classification

## D.1 Experiments on Imbalanced Point Cloud Classification

**Dataset and Baselines** In addition to 1D text and 2D image, we further investigate the robustness of our method on 3D point cloud data, where we use the popular ModelNet10 [45]. We select

Table 11: Ablation study on SST-2 with learned weight net under different imbalance factors.

| Method | 100 : 1000 | 50 : 1000 | 20 : 1000 | 10 : 1000 |
|---|---|---|---|---|
| **Label + Whole** | 86.48±0.29 | 86.30±0.37 | 86.19±0.54 | 86.03±0.46 |
| **Feature + Whole** | 86.08±0.38 | 86.08±0.37 | 85.98±0.29 | 85.95±0.47 |
| **Combined + Whole** | 86.71±0.24 | 86.49±0.60 | 86.47±0.51 | 86.24±0.60 |
| **Combined + Whole** | 87.08±0.09 | 87.13±0.04 | 87.14±0.08 | 87.10±0.05 |
| **Combined + Prototype** | 87.12±0.03 | 87.11±0.01 | 87.09±0.07 | 87.12±0.01 |
| **Combined + Random sample** | 86.59±0.50 | 86.14±0.52 | 86.08±0.77 | 85.85±0.52 |

PointNet++ [46] as our backbone without other additional layers. We use 5 examples in each class for meta set. Specifically, we down-sample the original points to 512 followed by a random point dropout, scale and shift to process the input data, which is a common setting for point cloud classification. We conduct the same process on all the experiments to evaluate our method fairly. Since we are the first to conduct imbalance classification task on point cloud data, we consider following methods: (1) **Pointnet++ baseline**[4], a classic network used in point cloud classification [46]. (2) **Focal loss** [12]. (3) **LDAM-DRW** [17]. (4) **Logit Adjustment** [20].

**Experimental details and results on ModelNet10** The classification results of different methods are summarized in Table 12, where we evaluate our method from the aspects of instance accuracy (IA) and class accuracy (CA). We can see that ours outperforms all baselines on IA and CA. These experimental results further demonstrate the effectiveness of integrating OT loss into imbalanced point cloud classification, showing the generalization ability of our proposed method to data modalities.

### D.2 Experimental settings

For point cloud classification, we select *bathtub* as minority class and the rest as majority, where the number of training sample is 100 in every class. We train the baseline model from scratch where we use Adam optimizer with learning rate $1e^{-4}$. Besides, our minibatch size is 128 and epochs are 40. For the first and second stage, we still use Adam optimizer, where learning rate is $1e^{-4}$ in the 1-st stage, $\alpha$ learning rate $1e^{-4}$ and $\beta$ learning rate $5e^{-3}$. For both two stages, we keep the same number of epochs and minibatch size as them in training the baseline. For training and meta dataset, we choice 100 samples from every majority class in original training set, where we then random choice 5 samples from the rest training set to build our meta set. For test dataset, we random sample 40 instances in every class from the original test dataset to construct our balanced test dataset.

Table 12: Comparison of the proposed method and baselines trained with different proportions on ModelNet10 for multi-class classification.

| Method | 1:100 | | 4:100 | | 8:100 | | 10:100 | |
|---|---|---|---|---|---|---|---|---|
| | IA | CA | IA | CA | IA | CA | IA | CA |
| Pointnet++ Baseline[46] | 81.75±0.82 | 80.59±1.07 | 82.53±1.10 | 81.42±1.35 | 88.67±0.37 | 88.28±0.75 | 87.95±0.27 | 87.58±0.42 |
| Focal Loss 0.25 $\gamma = 2$ [12] | 82.09±0.68 | 80.74±0.68 | 87.17±0.71 | 86.39±1.01 | 87.44±1.43 | 87.49±0.43 | 87.67±0.78 | 87.11±0.21 |
| LDAM-DRW [17] | 79.91±0.76 | 77.56±1.13 | 80.07±1.15 | 78.19±1.29 | 80.35±0.93 | 77.65±0.93 | 80.13±0.31 | 77.61±2.76 |
| Logit Adjustment $\tau = 1.0$ [20] | 82.81±0.57 | 81.86±0.41 | 84.88±0.93 | **84.39±1.28** | 86.77±1.07 | 86.28±0.88 | 87.56±1.13 | 87.13±1.20 |
| **Our method (Weight vector)** | **83.71±0.11** | **81.97±1.59** | **85.15±1.01** | 83.80±0.54 | **89.73±0.32** | **89.09±0.38** | **90.40±0.17** | **90.16±0.56** |

Table 13: Settings of the training on ModelNet10 dataset.

| Dataset | Stage 1 | Stage 2 |
|---|---|---|
| ModelNet10 | Adam ($1e^{-4}$)<br>epochs:10<br>batch size:64 | Adam ($\alpha$ $1e^{-4}$) and Adam ($\beta$ $5e^{-3}$)<br>epochs:50<br>batch size:64 |

### D.3 Ablation study

Similarly, we execute an ablation on point cloud to investigate the influence of OT loss, which is shown Table 14. We can see that the best performance is obtained when we use the setting of weight

[4]https://github.com/yanx27/Pointnet_Pointnet2_pytorch

Table 14: Ablation study on ModelNet 10 under different imbalance factors.

| Method | 1:100 | | 4:100 | | 8:100 | | 10:100 | |
|---|---|---|---|---|---|---|---|---|
| | IA | CA | IA | CA | IA | CA | IA | CA |
| **Weight Vector + Label + Whole** | 81.36±0.90 | 80.13±0.77 | 82.53±1.00 | 81.61±1.07 | 86.95±0.88 | 86.35±1.02 | 87.33±0.62 | 86.76±0.67 |
| **Weight Vector + Feature + Whole** | 81.81±0.86 | 80.59±0.87 | 83.20±0.40 | 82.16±0.34 | 87.00±0.83 | 86.35±0.64 | 88.06±0.40 | 87.43±0.14 |
| **Weight Vector + Combined + Prototype** | 83.04±0.35 | 81.97±0.32 | 85.16±1.28 | 84.11±1.10 | 89.73±0.69 | 89.46±0.64 | 89.84±0.19 | 89.50±0.41 |
| **Weight Vector + Combined + Whole** | 83.71±0.11 | 81.97±1.59 | 84.15±1.01 | 83.80±0.54 | 89.73±0.32 | 89.09±0.38 | 90.40±0.17 | 90.16±0.56 |
| **Weight Net + Combined + Whole** | 82.81±0.63 | 81.59±1.07 | 83.37±0.33 | 81.89±0.41 | 89.56±0.48 | 88.81±0.86 | 89.84±0.73 | 88.86±0.46 |

vector, combined cost and whole meta set. Besides, performance are also acceptable when we use label cost and feature cost. The performance is also competitive when we use the setting of weight vector, combined cost and prototype-based meta set.

# E Effect of Different Meta-sets

To explore whether our proposed method is effective even without an additional meta set from the validation set, we build a dynamic meta set from the given imbalanced training set. Notably, "dynamic" means that we randomly choose the samples from the training set to build a meta set at each training stage. We report experiment results in SST-2 dataset in Table 15. The results show our method could still reach a comparable performance even with the meta set built from the imbalance training dataset dynamically.

Table 15: Comparison of different meta set in SST-2 dataset, where additional meta set os built from the validation dataset and dynamic meta set is built from the imbalance training dataset directly.

| Method | 100 : 1000 | 50 : 1000 | 20 : 1000 | 10 : 1000 |
|---|---|---|---|---|
| **BERT Baseline (Without Fine Tuned)** | 74.91±4.62 | 53.26±5.70 | 50.54±1.40 | 49.84±0.02 |
| **BERT Baseline (Fine Tuned)** | 81.64±3.79 | 75.53±1.90 | 65.23±3.91 | 60.61±5.00 |
| **Baseline** | 78.25±2.24 | 57.18±1.88 | 55.00±1.23 | 50.17±1.34 |
| **Proportion** † | 79.59±3.35 | ✗ | 78.76±2.40 | 60.63±13.13 |
| **Hu et al.'s** ∗ **[1]** | 81.57±0.74 | 79.35±2.59 | 73.61±11.86 | 55.84±11.84 |
| **Hu et al.'s+Regularization** ∗ **[1]** | 82.25±1.16 | ✗ | 79.53±1.64 | 66.68±13.99 |
| **Liu et al.'s** ∗**[2]** | 82.58±0.98 | ✗ | 81.14±1.25 | 80.62±0.93 |
| **Logit Adjustment** $\tau = 1.0$ **[20]** | 79.23±1.68 | 71.19±2.09 | 56.16±15.45 | 49.50±17.11 |
| **LDAM-DRW [17]** | 78.89±4.89 | 66.30±13.92 | 65.17±10.88 | 53.89±17.70 |
| **Our method with additional meta set** | **87.08±0.09** | **87.13±0.04** | **87.14±0.08** | **87.10±0.05** |
| **Our method with dynamic meta set** | **87.12±0.03** | **87.11±0.01** | **87.09±0.07** | **87.12±0.01** |

# F Label-awareness Experiments

In this section, we report label-aware performance of our method in image, text and point cloud classification task. In this setting, **label-aware** represents that we only use label information in meta dataset to constrain the OT loss and then obtain the updated sample weights. **Our best performance** represents that we use both feature and label information in meta dataset. As summarized in 16, 17 and 18, all results show that our method could obtain a comparable performance under the **label-aware** setting, where this label information is free to be obtained and we could reach the goal by only requiring the labels in training dataset obeying a uniform distribution.

Table 16: Top-1 errors(%). Label-aware performance of our method in CIFAR-LT-100 dataset with imbalance factor=100, which is image classification task.

| Method | |
|---|---|
| **Our best performance** | 54.97 |
| **Label-aware** | 55.06 |

Table 17: Top-1 accuracy(%).Label-aware performance of our method in SST-2 dataset under different imbalance factors, which is text classification task.

| **Method** | 100 : 1000 | 50 : 1000 | 20 : 1000 | 10 : 1000 |
|---|---|---|---|---|
| **Our best performance** | 87.08±0.09 | 87.13±0.04 | 87.14±0.08 | 87.10±0.05 |
| **Label-aware** | 86.48±0.29 | 86.30±0.37 | 86.19±0.54 | 86.03±0.46 |

Table 18: Top-1 instance accuracy **IA** and class accuracy **CA**(%).Label-aware performance of our method in ModelNet 10 dataset under different imbalance factors, which is point cloud classification task.

| **Method** | 1:100 | | 4:100 | | 8:100 | | 10:100 | |
|---|---|---|---|---|---|---|---|---|
| | IA | CA | IA | CA | IA | CA | IA | CA |
| **Our best performance** | 83.71±0.11 | 81.97±1.59 | 85.15±1.01 | 83.80±0.54 | 89.73±0.32 | 89.09±0.38 | 90.40±0.17 | 90.16±0.56 |
| **Label-aware** | 81.36±0.90 | 80.13±0.77 | 82.53±1.00 | 81.61±1.07 | 86.95±0.88 | 86.35±1.02 | 87.33±0.62 | 86.76±0.67 |

# G Negative Societal Impacts

This paper introduces a re-weighting method based on optimal transport to solve the imbalanced problem. The proposed re-weighting method is very different from existing re-weighting methods and may have the potential to provide new and better thoughts for imbalanced problems in the future. Notably, once a malicious imbalanced task is selected, our re-weighting method may produce a negative societal impact, similar to the progress in other machine learning methods.