# A Semi-relaxed Optimal Transport

In this section, we first discuss the advantages of our formulation. We then demonstrate how to solve the semi-relaxed optimal transport using an efficient scaling algorithm. Finally, we analyze the differences between our algorithm and [7] in detail.

## A.1 Analysis of our OT formulation

Without loss of generality, we consider our formulation as the following generic form:

$$\min_{\mathbf{Q}} \langle \mathbf{Q}, \mathbf{C} \rangle_F + \gamma KL(\mathbf{Q}^\top \mathbf{1}_M, \mu) \tag{1}$$

$$\text{s.t. } \mathbf{Q} \in \{\mathbf{Q} \in \mathbb{R}^{M \times N} | \mathbf{Q}\mathbf{1}_N = \nu\}, \tag{2}$$

where $\mu, \nu$ are the prior marginal distribution of $\mathbf{Q}$, and $\mathbf{C}$ is the cost matrix. As analyzed in Appendix G.1, we observe that the imbalanced property of the dataset is reflected in the model's prediction $P$. OT generates pseudo labels $Q$ based on $P$ and several distribution constraints. Unlike typical OT, which imposes two equality constraints and enforces a uniform distribution, our relaxed OT utilizes a relaxed KL constraint on cluster size. In optimizing our relaxed OT, the optimal $Q$ for $< Q, -\log P >$ is assigned based on the largest prediction in $P$, capturing its imbalanced property. The KL constraints ensure that the marginal distribution of $Q$ remains close to the prior uniform distribution, avoiding degenerate solutions while providing flexibility. Consequently, the optimal $Q$ in our relaxed OT accounts for both the inherent imbalanced distribution of classes in $P$ and the prior uniform distribution, generating pseudo labels that reflect the imbalanced characteristics of $P$.

## A.2 Efficient Solver

While the above formulation suits our problem, its quadratic time complexity is unaffordable for large-scale problems. To solve this efficiently, motivated by [2], we first introduce an entropic constraint, $-\epsilon\mathcal{H}(\mathbf{Q})$. Due to

$$\langle \mathbf{Q}, \mathbf{C} \rangle_F - \epsilon\mathcal{H}(\mathbf{Q}) = \epsilon \langle \mathbf{Q}, \mathbf{C}/\epsilon + \log \mathbf{Q} \rangle_F = \epsilon \langle \mathbf{Q}, \log \frac{\mathbf{Q}}{e^{-\mathbf{C}/\epsilon}} \rangle_F, \tag{3}$$

The entropic semi-relaxed optimal transport can be reformulated as:

$$\epsilon \langle \mathbf{Q}, \log \frac{\mathbf{Q}}{e^{-\mathbf{C}/\epsilon}} \rangle_F + \gamma KL(\mathbf{Q}^\top \mathbf{1}_M, \mu) \tag{4}$$

$$\text{s.t. } \mathbf{Q} \in \{\mathbf{Q} \in \mathbb{R}^{M \times N} | \mathbf{Q}\mathbf{1}_N = \nu\}. \tag{5}$$

This problem can then be approximately solved by an efficient scaling algorithm. For more details, refer to [1].

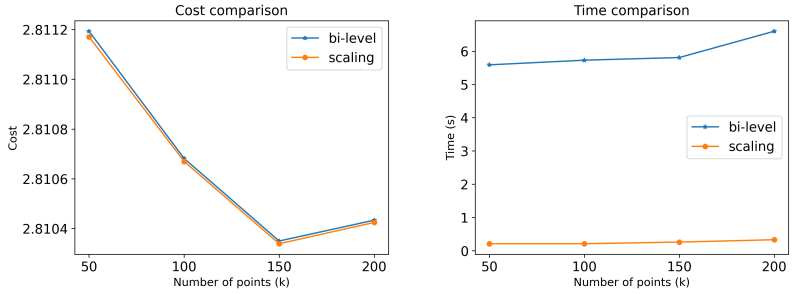



**Fig. 1:** Comparison of two optimization algorithms.

### A.3 Comparison with [7]

Zhang et al. [7] introduce an imbalanced self-labeling learning framework to tackle the issue of novel class discovery in long-tailed scenarios. To generate imbalanced pseudo-labels, they introduce an auxiliary variable $\mathbf{w} \in \mathbb{R}^N$, which is dynamically inferred during learning and encodes constraints on the cluster-size distribution. Their formulation is as follows:

$$\min_{\mathbf{Q}} \langle \mathbf{Q}, \mathbf{C} \rangle_F + \gamma KL(\mathbf{w}, \mu) \tag{6}$$

$$\text{s.t. } \mathbf{Q} \in \{\mathbf{Q} \in \mathbb{R}^{M \times N} | \mathbf{Q}\mathbf{1}_N = \nu, \mathbf{Q}^\top \mathbf{1}_M = \mathbf{w}\}, \tag{7}$$

Unlike our approach, they adopt a fixed $\gamma$. To optimize Equ.(6), they propose a bi-level optimization strategy, alternately estimating cluster distributions and generating pseudo labels by solving an optimal transport problem. Specifically, they start with a fixed $\mathbf{w}$ and first minimize Equ.(6) with respect to $\mathbf{Q}$. Since the KL constraint term remains constant, this task reduces to a standard optimal transport problem, which can be efficiently solved using the Sinkhorn-Knopp Algorithm. Then, they optimize Equ.(6) with respect to $\mathbf{w}$ using simple gradient descent.

While their bi-level optimization approximates the objective function, it consumes significantly more time compared to the direct application of the light-speed scaling algorithm. Additionally, it introduces extra hyperparameters $\mathbf{w}$ for inner-loop optimization. As shown in Fig. 1, the scaling algorithm is faster compared to the bi-level optimization strategy proposed by [7], making it feasible to solve large-scale problems.

## B Pseudo Code of our method

We provide a detailed description of our method in Alg.1 for clarity and have released our code in https://github.com/RikkiXu/NCD_PC.

**Algorithm 1:** Dual-level Imbalanced-aware Self-labeling and Learning Algorithm

**Input:** Trainset $\mathcal{D}$, softmax function $\sigma$, encoder $f_\theta$, classifier $h = [h^s, h^u] \in \mathbb{R}^{D\times(|C^s|+|C^u|)}$, $\mu = \mathbf{1}$, uniform distribution $\nu$, hyperparameters $\gamma_p, \gamma_r, \lambda, \rho, T, \alpha, \beta, \epsilon$

**for** $e \in 1, 2, .., Epoch$ **do**

  **for** $s \in 1, 2, ..., Step$ **do**

    $\{(x_i^s, y_i^s)\}_{i=0}^N, \{x_j^u\}_{j=0}^M \leftarrow \text{Sample}(\mathcal{D})$

    `//Cluster point into different regions by DBSCAN`

    $\{r_k\}_{k=0}^K \leftarrow DBSCAN(\{x_j^u\}_{j=0}^M)$

    `//Forward the model`

    $p^s = (h^s \circ f_\theta(x^s)/\tau)$

    $\mathbf{z}_p = f_\theta(x^u), \mathbf{z}_r = \{\text{AvgPool}(\mathbf{z}_p)|r_k \text{ is same}\}$

    $p_p^u = (h^u \circ \mathbf{z}_p/\tau), p_r^u = (h^u \circ \mathbf{z}_r/\tau)$

    `//CE loss for known classes`

    $\mathcal{L}_s = -\frac{1}{N}\sum_{i=1}^N y_i^s \log p_i^s$

    `//Point level self-labeling`

    $\mathbf{Q}_p^u = -\log \mathbf{P}_p^u,\ \gamma_t^p,\ \epsilon$

    $\mathcal{L}_p^u = \frac{1}{M}\langle \mathbf{Q}_p^u, -\log \mathbf{P}_r^u \rangle_F$

    `//Region level self-labeling`

    $\mathbf{Q}_r^u = -\log \mathbf{P}_r^u,\ \gamma_t^r,\ \epsilon$

    $\mathcal{L}_u^r = \frac{1}{K}\langle \mathbf{Q}_r^u, -\log \mathbf{P}_r^u \rangle_F$

    `//Update` $\gamma_{t+1}^p$ `and` $\gamma_{t+1}^r$

    **if** $KL(\frac{1}{M}\mathbf{Q}_p^{u\top}\mathbf{1}_M, \nu) \leq \rho$ *consistently for* $T$ *iterations* **then**

      $\gamma_{i+1}^p = \lambda\gamma_i^p$

    **end**

    **if** $KL(\frac{1}{K}\mathbf{Q}_r^{u\top}\mathbf{1}_M, \nu) \leq \rho$ *consistently for* $T$ *iterations* **then**

      $\gamma_{i+1}^r = \lambda\gamma_i^r$

    **end**

    `//Total loss`

    minimize $\mathcal{L}_s + \alpha\mathcal{L}_u^p + \beta\mathcal{L}_u^r$ *w.r.t* $\theta$

  **end**

**end**

## C Dataset Splits

We follow [6] and partition SemanticKITTI and SemanticPOSS into four splits, detailed in Tab. 1 and 2. It's important to note that [6] balance the distribution of novel classes across splits to prevent the most frequent novel class from biasing other classes and to leverage semantic relationships between known and novel classes. The left and middle plots in Fig. 2 illustrate the distributions of SemanticKITTI and SemanticPOSS across these splits. However, this deliberate selection of novel classes may not adequately address point cloud data imbalance.

To assess the generalization of our algorithm, we conduct experiments on a more challenging benchmark. Here, we select half of the classes from Seman-

**Table 1:** The detail of novel classes in each split.

| Split | SemanticKITTI | SemanticPOSS |
|---|---|---|
| 0 | building,road,sidewalk,terrain,vegetation | building,car,ground,plants |
| 1 | car,fence,other-ground,parking,trunk | bike,fence,person |
| 2 | motorcycle,other-vehicle,pole,traffic-sign,truck | pole,traffic-sign,trunk |
| 3 | bicycle,bicyclist,motorcyclist,person | cone-stone,rider,trashcan |

**Table 2:** The detail of novel classes on the challenging setting of SemanticPOSS dataset.

| Split | SemanticPOSS |
|---|---|
| 0 | building,car,ground,plants,bike,fence,person |
| 1 | pole,traffic-sign,trunk,cone-stone,rider,trashcan |

ticPOSS as novel classes, as depicted on the right side of Fig. 2. Results and analysis are presented in Section 4.2.

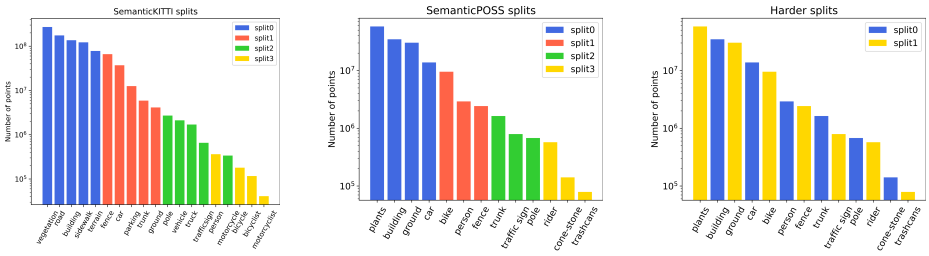



**Fig. 2:** Distribution plot of the SemanticPOSS dataset. Each class has been assigned the color of the split which has to be considered novel.

## D Analysis on Detailed Results

To validate our method's ability to handle imbalanced data distributions, we compute the mIoU for head classes, medium classes, and tail classes within each split. The results for both datasets are presented in Tab. 3 and Tab. 4. Specifically, for SemanticPOSS, in the first split, there are a total of 4 novel classes. We choose the two classes with the highest count as head classes, and the remaining two are designated as medium and tail classes. In the other splits, which all have 3 novel classes, we sort them by size and assign them as head, medium, and tail classes accordingly. For SemanticKITTI, we designated the largest class and the smallest class within each split as head and tail classes, respectively, with the rest categorized as medium classes.

**Table 3:** Detailed results for SemanticPOSS

| Method | Head | Medium | Tail |
|---|---|---|---|
| NOPS | 37.5 | 21.9 | 4.4 |
| Ours | 45.0 | 30.8 | 11.2 |

**Table 4:** Detailed results for SemanticKITTI

| Method | Head | Medium | Tail |
|---|---|---|---|
| NOPS | 26.1 | 28.3 | 7.4 |
| Ours | 33.8 | 32.2 | 8.5 |

As shown in Tab. 3, in SemanticPOSS, our approach achieve improvements of 7.5%, 8.9%, and 6.8% for head, medium, and tail classes, respectively, compared to NOPS. Similarly, in SemanticKITTI, our method yielded improvements of 7.7%, 3.9%, and 1.1% for head, medium, and tail classes, respectively. These results demonstrate that our method effectively addresses imbalanced data scenarios by improving performance across all class categories. It's worth noting that NOPS utilizes additional techniques such as multi-head and overclustering during training to enhance its performance, whereas our method achieves these improvements without employing such techniques, highlighting its efficiency and effectiveness.

## E Comparison with NOPS variant

We observed that NOPS [6] employs a learning rate of 0.01 with SGD, which proved excessively low and caused training not to converge. To ensure convergence, we modified the optimization strategy by switching to the AdamW optimizer with an initial learning rate of 1e-3, gradually decreasing to 1e-5 following a cosine schedule. This adjustment successfully led to the convergence of NOPS during training. The results, as shown in Tab. 5 and Tab. 6, indicate that while the mIoU for known classes improves, there is a decrease in mIoU for novel classes as a consequence. Nevertheless, our method continues to outperform NOPS by a significant margin.

## F Analysis of Adaptive Regularization

We visualize the curve depicting changes in the pseudo-label distribution before and after adding adaptive regularization, as shown in Fig. 3. Initially, the pseudo-label distribution in Baseline+ISL remains consistently uniform, with each of the four classes occupying 25%. This indicates that the uniform constraint is too strong during later stages of training, leading to a pseudo-label distribution that tends towards uniformity, not reflecting the actual imbalanced point cloud data and resulting in suboptimal outcomes.After applying adaptive regularization, we dynamically adjust the uniform constraint based on the KL distance between the pseudo-label distribution and a uniform distribution. As depicted in the right plot of Fig. 3, the curves representing changes in the pseudo-label distribution for each class do not converge towards uniformity. This demonstrates that our adaptive

**Table 5:** NOPS* denotes NOPS with our training setting.

| Split | Method | bike | build. | car | cone. | fence | grou. | pers. | plants | pole | rider | traf. | trashc. | trunk | Novel | Known | All |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Full | 45.0 | 83.3 | 52.0 | 36.5 | 46.7 | 77.6 | 68.2 | 77.7 | 36.0 | 58.9 | 30.3 | 4.2 | 14.4 | - | - | 48.5 |
| 0 | NOPS | 35.5 | 30.4 | 1.2 | 13.5 | 24.1 | 69.1 | 44.7 | **42.1** | 19.2 | 47.7 | 24.4 | 8.2 | 21.8 | 35.7 | 26.6 | 29.4 |
| | NOPS* | 46.9 | 16.1 | 4.2 | 35.4 | 47.8 | 54.9 | 67.1 | 37.9 | 36.1 | 62.3 | 28.9 | 1.8 | 20.2 | 28.3 | 38.1 | 35.3 |
| | Ours | 46.5 | **70.3** | **7.6** | 31.1 | 49.4 | **82.2** | 67.1 | 41.7 | 37.5 | 57.5 | 29.8 | 8.4 | 14.1 | **50.4** | 37.9 | 41.8 |
| 1 | NOPS | 29.4 | 71.4 | 28.7 | 12.2 | 3.9 | 78.2 | **56.8** | 74.2 | 18.3 | 38.9 | 23.3 | 13.7 | 23.5 | 30.0 | 38.2 | 36.4 |
| | NOPS* | 21.5 | 83.2 | 49.9 | 29.7 | 20.0 | 77.7 | 29.6 | 77.3 | 37.4 | 58.0 | 26.2 | 3.4 | 15.5 | 23.7 | 45.8 | 40.7 |
| | Ours | **31.5** | 83.2 | 48.7 | 25.4 | **23.9** | 77.3 | 53.1 | 77.1 | 32.5 | 57.3 | 35.0 | 9.3 | 18.0 | **36.2** | 46.4 | 44.0 |
| 2 | NOPS | 37.2 | 71.8 | 29.7 | 14.6 | 28.4 | 77.5 | 52.1 | 73.0 | 11.5 | 47.1 | 0.5 | 10.2 | **14.8** | 9.0 | 44.2 | 36.0 |
| | NOPS* | 44.5 | 83.0 | 51.4 | 25.2 | 47.5 | 77.0 | 66.0 | 76.4 | **23.0** | 59.9 | 4.2 | 8.4 | 4.4 | 10.5 | 53.9 | 43.9 |
| | Ours | 45.3 | 82.8 | 49.8 | 28.4 | 46.3 | 76.7 | 66.2 | 77.2 | 10.9 | 58.4 | **18.6** | 7.3 | 8.2 | **12.6** | 53.8 | 44.3 |
| 3 | NOPS | 38.6 | 70.4 | 30.9 | 0.0 | 29.4 | 76.5 | 56.0 | 71.8 | 17.0 | 31.9 | 26.2 | 1.0 | 22.6 | 10.9 | 43.9 | 36.3 |
| | NOPS* | 44.4 | 83.1 | 46.9 | **0.2** | 43.0 | 77.9 | 65.2 | 78.0 | 34.8 | 45.3 | 32.4 | 1.9 | 14.5 | 15.8 | 52.0 | 43.7 |
| | Ours | 45.5 | 82.9 | 47.7 | 0.0 | 45.1 | 77.8 | 66.3 | 77.7 | 34.3 | **49.1** | 35.6 | **4.0** | 15.3 | **17.7** | 52.8 | 44.7 |

**Table 6:** The novel class discovery results on the SemanticKITTI dataset. 'Full' denotes the results obtained by supervised learning. The gray values are the novel classes in each split.

| Method | bi.cle | b.clst | build. | car | fence | mt.cle | m.clst | oth-g. | oth-v. | park. | pers. | pole | road | side2. | terra. | traff. | truck | trunk | veget. | Novel | Known | All |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Full | 2.9 | 55.4 | 89.5 | 93.5 | 27.9 | 27.4 | 0.0 | 0.9 | 19.9 | 35.8 | 31.2 | 60.0 | 93.5 | 77.8 | 62.0 | 39.8 | 50.8 | 53.9 | 87.0 | - | - | 47.9 |
| NOPS | 5.6 | 47.8 | 52.7 | 82.6 | 13.8 | 25.6 | 1.4 | 1.7 | 14.5 | 19.8 | 25.9 | 32.1 | **56.7** | 8.1 | **23.8** | 14.3 | 49.4 | 36.2 | 44.2 | 37.1 | 26.5 | 29.3 |
| NOPS* | 7.9 | 55.9 | 46.7 | 89.3 | 24.7 | 27.7 | 0.0 | 1.1 | 22.7 | 25.5 | 33.8 | 57.0 | 43.2 | 17.9 | 21.7 | 39.3 | 61.5 | 50.8 | 23.9 | 30.7 | 35.5 | 34.2 |
| Ours | 5.5 | 51.1 | **74.6** | 92.3 | 29.8 | 22.8 | 0.0 | 0.0 | 23.3 | 24.8 | 27.7 | 59.7 | 41.4 | **22.5** | 23.6 | 39.3 | 43.6 | 51.1 | **66.4** | **45.7** | 33.7 | 36.8 |
| NOPS | 7.4 | 51.2 | 84.5 | 50.9 | 7.3 | 28.9 | 1.8 | 0.0 | 22.2 | 19.4 | 30.4 | 37.6 | 90.1 | 72.2 | 60.8 | 16.8 | 57.3 | **49.3** | 85.1 | 25.4 | 46.2 | 40.7 |
| NOPS* | 2.1 | 52.3 | 89.2 | 52.3 | 6.5 | 27.1 | 0.0 | 0.0 | 18.4 | 17.5 | 33.1 | 59.3 | 90.2 | 77.2 | 61.9 | 39.9 | 53.1 | 19.5 | 86.7 | 19.2 | 49.8 | 41.9 |
| Ours | 3.7 | 57.4 | 89.2 | **56.5** | **17.3** | 20.3 | 0.0 | 0.0 | 20.0 | **30.6** | 34.8 | 60.6 | 93.2 | 77.6 | 62.0 | 38.7 | 56.9 | 39.2 | 86.7 | **28.7** | 50.1 | 44.5 |
| NOPS | 6.7 | 49.2 | 86.4 | 90.8 | 23.7 | 2.7 | 0.6 | 1.9 | **15.5** | 29.5 | 27.9 | **36.4** | 90.3 | 73.4 | 61.2 | **17.8** | 10.3 | 46.2 | 84.3 | 16.5 | 48.0 | 39.7 |
| NOPS* | 4.1 | 55.2 | 89.1 | 93.4 | 29.1 | 0.6 | 0.0 | 0.5 | 2.8 | 33.9 | 30.9 | 32.7 | 93.1 | 77.7 | 60.9 | 0.1 | 32.9 | 52.2 | 86.4 | 13.8 | 50.3 | 40.8 |
| Ours | 3.6 | 54.2 | 88.9 | 93.3 | 28.4 | **10.2** | 0.0 | 0.9 | 9.6 | 33.4 | 32.2 | 36.1 | 92.7 | 77.4 | 62.2 | 10.7 | **34.2** | 51.7 | 86.9 | **20.1** | 50.4 | 42.5 |
| NOPS | 2.3 | 27.8 | 86.0 | 89.9 | 23.1 | 24.5 | **2.9** | 3.1 | 18.2 | 30.1 | **16.3** | 39.9 | 90.7 | 73.5 | 61.0 | 17.4 | 49.8 | 44.0 | 83.2 | 12.4 | 49.0 | 41.2 |
| NOPS* | 2.3 | 16.9 | 89.5 | 93.9 | 28.2 | 27.5 | 0.0 | 0.6 | 25.3 | 34.2 | 3.1 | 60.4 | 93.2 | 77.7 | 61.3 | 38.9 | 67.0 | 54.4 | 86.6 | 5.6 | 55.9 | 45.3 |
| Ours | **2.6** | **32.5** | 88.7 | 93.3 | 28.1 | 24 | 0.1 | 1.0 | 23.7 | 35.6 | 15.3 | 59.8 | 93.2 | 77.6 | 61.4 | 37.8 | 56.6 | 52.1 | 86.7 | **12.6** | 54.6 | 45.8 |

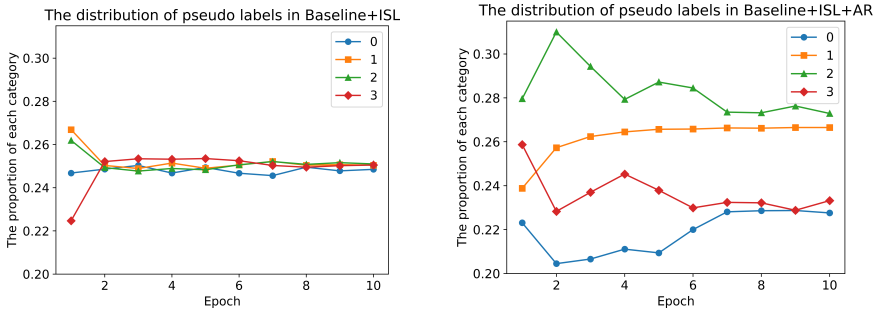



**Fig. 3:** The pseudo label distribution before and after adding adaptive regularization

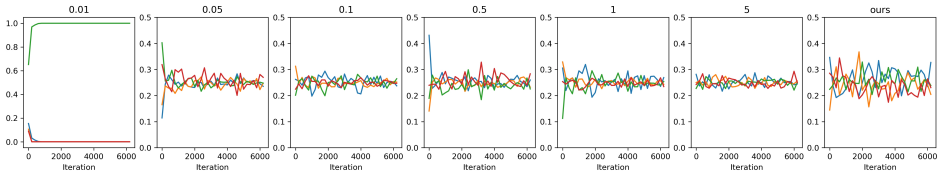



**Fig. 4:** Class distribution for different fixed $\gamma$ during the training.

regularization, compared to a fixed $\gamma$, provides more flexibility in learning a pseudo-label distribution that better aligns with imbalanced point cloud data.

To illustrate how $\gamma$ affects the pseudo-label distribution, we present the class distribution of four classes for different fixed $\gamma$ values during training. As shown in Fig. 4, we observe the following: 1. A small $\gamma$ leads to a degenerate solution. 2. Increasing $\gamma$ gradually pushes the distribution towards uniformity. 3. Our adaptive $\gamma$ approach maintains flexibility, resulting in an imbalanced class distribution.

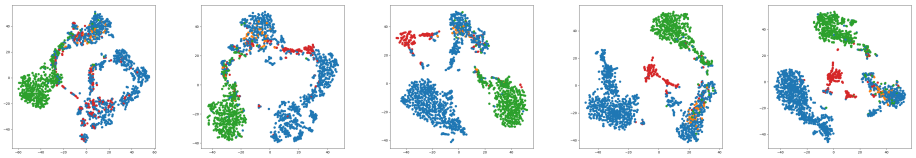
## G More Analysis of Prototype

### G.1 Initialization and updating process

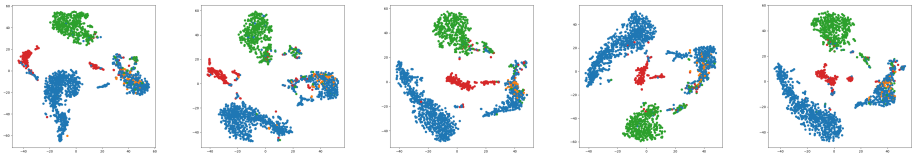
In the beginning, the representation and prototypes are randomly initialized, which is very noisy. However, there are three key factors that guarantee us to gradually improve the representation and prototype. The first one is the learning of seen classes, which improves the representation ability of our model, thus improving the representation of novel classes implicitly. To prove that known classes can help the representation of novel classes, we cluster the representation of novel classes obtained from a known-class supervised pre-trained model and a randomly initialized model on SemanticPOSS split 0.

The results indicate that features extracted from a known-class pre-trained model exhibit better clustering performance compared to features extracted from a randomly initialized model. The former outperforms the latter by nearly 7% in mIoU for novel classes, demonstrating that known classes can indeed enhance the representation of novel classes. The second one is the view-invariant training, which learns invariant representation for different transformations and promotes the representation directly. Some studies [5, 8] have advanced unsupervised representation learning for point clouds by incorporating transformation invariance. The third one is the utilization of spatial prior, which enforces the point in the same region to be coherent, which may be validated by Fig.3 and 4, and unsupervised clustering [5, 9].

Those factors gradually improve the representation and prototype, leading to an informative prediction $P$. Then, our adaptive self-labeling algorithm utilizes $P$ and several marginal distribution constraints to generate pseudo-label $Q$. Finally, the $Q$ guides the learning of representation and prototype. In conclusion, the above three factors and our self-labeling learning process ensure our method



**(a)** After 1 epoch **(b)** After 2 epochs **(c)** After 3 epochs **(d)** After 4 epoch **(e)** After 5 epochs



**(f)** After 6 epochs **(g)** After 7 epoch **(h)** After 8 epochs **(i)** After 9 epochs **(j)** After 10 epoch

**Fig. 5:** The quality of the representation of unseen classes during training in SemanticPOSS split 0. Blue dots are plants, green dots are the ground, orange are cars, and red are buildings

**Table 7:** Ablation alternate design of region-level prototype on split 0 of SemanticPOSS dataset. The results are on novel classes.

| Prototype Sharing | Building | Car | Ground | Plants | Avg |
|---|---|---|---|---|---|
| × | 25.4 | 9.5 | 81.6 | 31.0 | 36.9 |
| ✓ | 51.5 | 6.0 | 83.0 | 53.1 | 48.4 |

learns meaningful representation and prototypes gradually. Furthermore, we visualize the representation of novel classes during training in Fig. 5, showing that as the training time increases, the learned representation gradually becomes better, validating our analysis.

### G.2 Prototype sharing of region-level learning

We conduct experiments without sharing prototypes, and the results are depicted in Tab. 7. It is noteworthy that utilizing two isolated prototypes results in a significant drop of nearly 10% in performance in the novel class.

In addition, when prototypes are not shared, we visualize the similarity matrix between the two prototypes. As shown in Fig. 6, we observe that the similarity between the two prototypes is low, which validates that having two prototypes leads to disparities in point-wise and region-wise learning directions. In contrast, sharing a prototype avoids this issue.

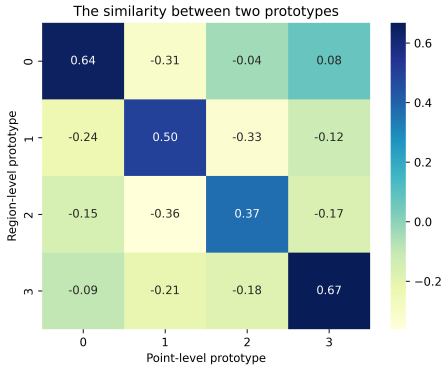



**Fig. 6:** The similarity of two prototypes

**Table 8:** More ablation experiments on SemanticPOSS

| Aug | Two views | ISL+AR+Region | Building | Car | Ground | Plants | mIoU |
|---|---|---|---|---|---|---|---|
| | | ✓ | 22.1 | 2.1 | 34.4 | 24.8 | 20.9 |
| ✓ | | ✓ | 46.3 | 1.9 | 34.5 | 18.6 | 25.3 |
| ✓ | ✓ | | 21.6 | 2.7 | 76.6 | 26.1 | 31.8 |
| ✓ | ✓ | ✓ | 51.5 | 6.0 | 83.0 | 53.1 | 48.4 |

## H More details on augmentation

Using augmentation to create two views is a well-established technique for learning a transformation-invariant representation, widely employed in novel class discovery literature [3,4], and more recently applied in point clouds [5,6,9]. In our comparison, the previous sota method [6] also adopts the same augmentation as ours to generate two views. Therefore, our comparison ensures a fair assessment.

To show the effect of augmentation, we conduct ablation on the two views and augmentation.

As Tab. 8, From the results above, we draw the following conclusion: 1) Compare columns 1 and 2, augmentation on one view has improved by 4.4% in novel classes compared with no augmentation; 2) Compare columns 2 and 4, employing two views has improved by 23.1% in novel classes; 3) Compare column 3 and 4, our proposed techniques result in a significant improvement of 17%.

## I More ablation study

We conduct additional ablation on split 0 of SemanticKITTI. The results are shown in Tab. 9: Similar to SemanticPOSS, each component enhances performance. Specifically, adaptive regularization and region-level learning individually contribute to a 6.6% and 5.0% improvement in mIoU for the model.

**Table 9:** More ablation experiments on SemanticKITTI split 0

| ISL | AR | Region | Building | Road | Sidewalk | Terrain | Vegetation | Avg |
|---|---|---|---|---|---|---|---|---|
| | | | 46.7 | 43.2 | 17.9 | 21.7 | 23.9 | 30.7 |
| ✓ | | | 57.4 | 32.1 | 25.2 | 18.9 | 37.2 | 34.1 |
| ✓ | ✓ | | 70.8 | 34.7 | 23.2 | 16.8 | 57.9 | 40.7 |
| ✓ | ✓ | ✓ | 74.6 | 41.4 | 22.5 | 23.6 | 66.4 | 45.7 |

# J More details on DBSCAN

## J.1 Parameters for the DBSCAN algorithm

DBSCAN is a density-based clustering algorithm: given a set of points in some space, it groups points close to each other, marking as outliers points that lie alone in low-density regions. DBSCAN has two key parameters: epsilon and min-samples. epsilon represents the maximum distance between two samples for one to be considered as in the neighborhood of the other, while min-samples denote the minimal number of samples in a region. In our experiments, we set min-samples to be reasonable minimal 2, indicating that there must be at least two points in a region. For epsilon, we determine a value of 0.5 based on the proportion of outliers, ensuring that 95% of the point clouds participate in region branch learning. In the following part, we conduct experiments with different epsilon values and analyze the results.

## J.2 Visual examples of the resultant regions

We present visualizations of regions under different epsilon values in Fig. 7. As shown in the visualizations, a smaller epsilon results in more outliers and smaller generated regions. Conversely, a higher epsilon leads to fewer outliers and larger generated regions.

## J.3 Sensitivity analysis of DBSCAN parameters

As shown in the Tab. 10, we supplement the proportion of outlier points in the 7th column and model training results under different epsilon in the 6th column. To assess the quality of regions, we assign a category label to each region based on the category with the highest point count within the region, with outliers being disregarded, and then calculate the mIoU in the 8th column. The results indicate that selecting 0.5 based on the outlier ratio yields satisfactory outcomes. Moreover, fine-tuning epsilon, for instance, setting it to 0.7, leads to improved performance. It is worth noting that the results first increased and then decreased with the increase of epsilon. This is because when epsilon is low, as shown in the visualization, there are more outliers, the generated region is smaller, and less spatial context information is used. When epsilon is higher, the generated

**Table 10:** Model training results, proportion of outlier points, and region mIoU under different epsilon. Region mIoU is the mIoU between regions label and ground true. The region label is composed of each region label, which is the category with the most points in each region. Region mIoU ignores outliers.

| epsilon | Building | Car | Plants | Ground | mIoU | Outlier | Region mIoU |
|---|---|---|---|---|---|---|---|
| 0.1 | 41.5 | 1.7 | 45.6 | 80.7 | 42.4 | 45.6% | 97.0 |
| 0.3 | 49.2 | 8.3 | 49.2 | 83.8 | 47.6 | 7.5% | 84.5 |
| 0.5 | 51.5 | 6.0 | 53.1 | 83.0 | 48.4 | 2.5% | 74.8 |
| 0.7 | 65.5 | 9.0 | 61.3 | 78.2 | 53.5 | 1.3% | 64.5 |
| 1 | 49.2 | 8.9 | 55.3 | 82.9 | 49.1 | 0.5% | 44.3 |

region is larger and the Regions mIoU is lower, resulting in noisy region-level representation.

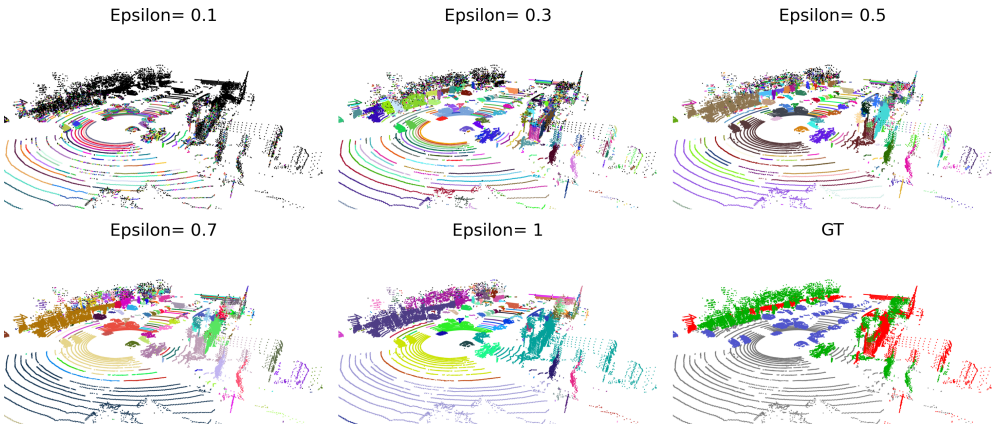



**Fig. 7:** Visualization of regions under different epsilon. For the first five pictures, the black point clouds are outliers, which means that the point does not belong to any region. Other random colors represent a region.

## K More Visualization

To demonstrate the effectiveness of our method, we created a video to compare NOPS with our prediction results, and our method shows a significant improvement over NOPS.

## References

1. Chizat, L., Peyré, G., Schmitzer, B., Vialard, F.X.: Scaling algorithms for unbalanced optimal transport problems. Mathematics of Computation **87**(314), 2563–2609 (2018)

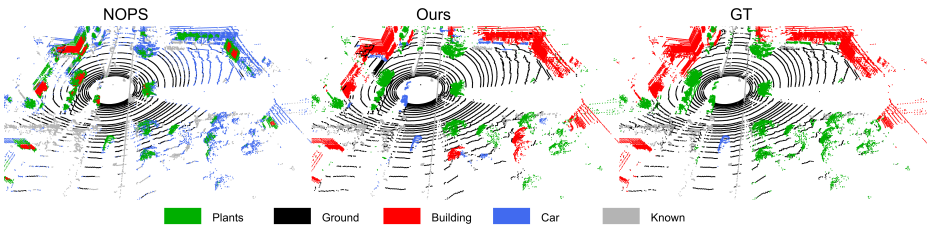



**Fig. 8:** One frame from our video, the dataset being SemanticPOSS split 0.

2. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. Advances in neural information processing systems **26** (2013)
3. Fini, E., Sangineto, E., Lathuilière, S., Zhong, Z., Nabi, M., Ricci, E.: A unified objective for novel class discovery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9284–9292 (2021)
4. Han, K., Rebuffi, S.A., Ehrhardt, S., Vedaldi, A., Zisserman, A.: Autonovel: Automatically discovering and learning novel visual categories. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
5. Long, F., Yao, T., Qiu, Z., Li, L., Mei, T.: Pointclustering: Unsupervised point cloud pre-training using transformation invariance in clustering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21824–21834 (2023)
6. Riz, L., Saltori, C., Ricci, E., Poiesi, F.: Novel class discovery for 3d point cloud semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9393–9402 (2023)
7. Zhang, C., Xu, R., He, X.: Novel class discovery for long-tailed recognition. Transactions on Machine Learning Research (2023)
8. Zhang, Z., Girdhar, R., Joulin, A., Misra, I.: Self-supervised pretraining of 3d features on any point-cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10252–10263 (2021)
9. Zhang, Z., Yang, B., Wang, B., Li, B.: Growsp: Unsupervised semantic segmentation of 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17619–17629 (2023)