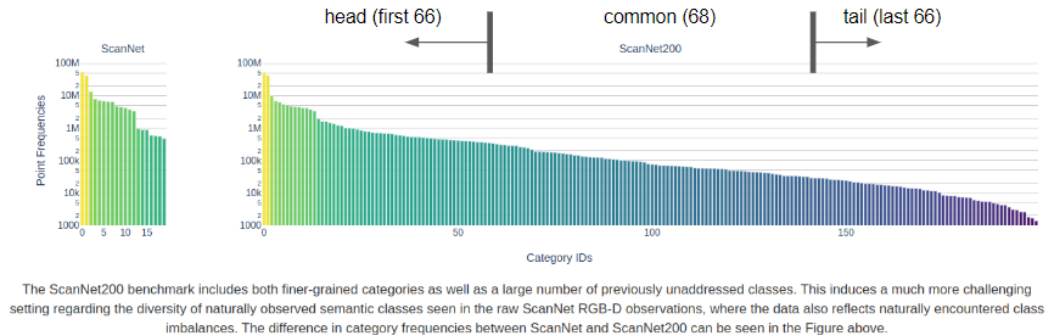


# 3DCV Final Project-- ScanNet200

## 3D Long-Tail Indoor Scene Semantic Segmentation

Team1 r11943018 林子軒、r11942101 周子庭、r11943022 范詠為



### 1 Introduction

3D long-tail indoor scene semantic segmentation is the task of assigning a semantic label (such as "wall," "floor," "chair," etc.) to each voxel (a 3D pixel) in a 3D indoor scene. This task is important because it allows a computer to understand the layout and contents of an indoor space, which has a wide range of applications including virtual reality, robotics, and automated building management.

One challenge in 3D long-tail indoor scene semantic segmentation is the "long-tail" distribution of classes in the data. This means that there are a large number of classes, but the number of examples for each class is highly imbalanced. For example, there may be many more examples of "wall" than there are of "vase" or "potted plant." This can make it difficult for a model to accurately classify the rarer classes.

To address this challenge, it is often necessary to use techniques such as data augmentation, class balancing, and transfer learning to improve the model's performance on the rarer classes. It is also important to use a robust evaluation metric that takes into account the imbalanced class distribution, such as mean intersection over union(mIOU).

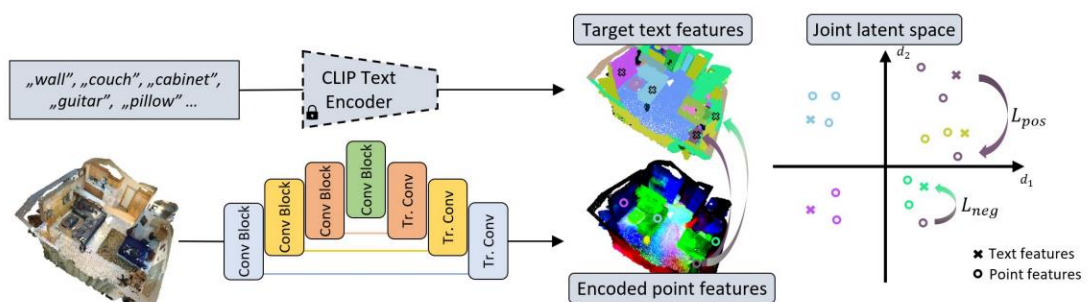


Fig. 1 Model architecture

## 2 Methodology

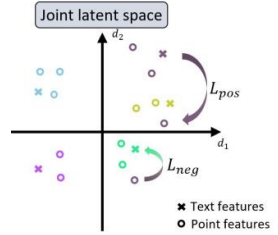
We take 2022 ECCV paper, Language-Grounded Indoor 3D Semantic Segmentation in the Wild as reference, which is also the paper proposed the first 3D indoor long-tail dataset, ScanNet 200.(see Fig. 1)

Their method is quite simple. They tackle the task of assigning a semantic label to each voxel in a 3D indoor scene from the ScanNet dataset, which contains 200 classes. It uses language models that have been trained on a large number of observations across all of the classes to exploit their well-structured nature. Pre-trained text embeddings from CLIP are used as anchors to help map geometric features during a pre-training step. These language-grounded features are able to build the relation between different class label, making rare class can be benefit from common class utilizing their text relationship.

- 3D UNet Backbone

First, a 3D sparse convolution UNet (specifically, we used Minikowiski Engine), was used as the 3D feature extraction backbone. This UNet is a type of neural network architecture that is commonly used for segmentation tasks. It is designed to be able to extract high-level features from input data, and in this case, it is specifically used to generate per-voxel semantic features from 3D indoor scenes. The UNet architecture consists of an encoder structure, which captures context and spatial information, and a decoder structure, which combines this information with the high-level features to produce a dense prediction. The use of the 3D sparse convolution UNet as the feature extraction backbone allows the model to efficiently and effectively capture and use detailed information about the 3D structure and contents of the indoor scenes.

$$\mathcal{L}_{pos} = \sum_{i=1}^{N_p} \max \left( 0, \frac{f_i^s \cdot f_{h(i)}^t}{|f_i^s| \cdot |f_{h(i)}^t|} - t_{pos} \right)$$

$$\mathcal{L}_{neg} = \sum_{i=1}^{N_p} \frac{1}{|M|} \sum_{j \in M} \max \left( 0, t_{neg} - \frac{f_i^s \cdot f_j^t}{|f_i^s| \cdot |f_j^t|} \right)$$


The diagram illustrates the 'Joint latent space' with axes  $d_1$  and  $d_2$ . It shows two sets of features: 'Text features' represented by 'x' marks and 'Point features' represented by 'o' marks. A purple arc labeled  $L_{pos}$  indicates the positive loss region where text and point features are pulled together. A green arc labeled  $L_{neg}$  indicates the negative loss region where text and point features are pushed apart. A legend at the bottom right identifies the symbols: 'x Text features' and 'o Point features'.

Fig. 2 Contrastive Loss

- Contrastive Loss

Second, the 3D UNet is trained to map to the well-structured space of the text model(CLIP) by using a contrastive objective to bring together different data modalities. The objective is defined using a batch of 3D scans, each of

which has sparse voxel locations. The 3D features for each voxel are mapped to text features representing the semantic label of the voxel. Additionally, multiple non-matching semantic text features, sampled from all text semantic labels, are pushed away from the learned features as negatives. Using cosine distance between the feature allows for more flexibility in the feature learning by constraining only vector directions, similar to the approach used in CLIP-driven image classification. The final language-3D pre-training objective is a combination of the positive and negative loss terms, where the effect of the negative terms is weighted by a factor of  $\lambda$  (see Fig. 2). Empirically, it was found that negative sampling was necessary for effective 3D representation learning, rather than using positive text associations only.

- Our Improvement

Based on above method from the 2022 ECCV paper, we already got quite decent result. We still find some way to improve the model:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t).$$

- i. Focal Loss

As shown above, Focal Loss is a variant of Cross-Entropy Loss that is designed to address the issue of class imbalance in a dataset. It does this by assigning higher weights to hard or easily misclassified examples, such as background objects with noisy texture or partial objects, and lower weights to easy examples, such as background objects. This means that Focal Loss puts more emphasis on correctly classifying the hard or misclassified examples, while down-weighting the importance of correctly classifying the easy examples. The goal of this is to improve the model's overall performance by focusing more on the examples that are most likely to be misclassified, which is perfectly match our goal to more accurately classify those rare class objects.



Guitar



Ball



Water Bottle



Keyboard Piano

*Fig. 3 Tail instance*

- ii. Tail Instance Sampling

Next, rare class objects are resampled by first preprocessing and saving them as separate .ply files. During training, these files are randomly sampled and added back into a training scene, where they are placed on the floor to avoid collisions with other objects. This process is repeated for a total of 5 instances per training scene. The goal of this is to increase the number of these rare class objects in the training data, with the hope that the model will become more robust in predicting the rare classes. By adding these additional instances to the training data, the model is exposed to a larger variety of examples for the rare classes, which may help it to better generalize to new instances of these classes during inference.

### 3 Encountered Problem

The first problem is the computation consumption is insanely high when doing indoor voxel scene. When solely using the reference baseline model we need 2 RTX 3090 to train on their original setting, not to mention that it takes two days to converge. So, after that we change the voxel size into 0.04m (originally 0.02). The bigger the voxel size is, the lower the resolutions are. It means that the model performance is affected.

Second problem is the scope of the code. It is too large to handle and write everything by ourselves in three weeks. So, we only do the modification on the original GitHub repository of the 2022 ECCV paper.

#### 4 Experiment

We train the below experiment on 2 RTX TITAN with batch size 2 in voxel size 0.04.

**s1-s2-s3 each 200 epoch**

| mIoU (train)                          | Head   | Common | Tail   | All    | All (test) |
|---------------------------------------|--------|--------|--------|--------|------------|
| train best - ce - none                | 66.554 | 56.430 | 37.069 | 53.382 | 18.4486    |
| train best - focal - none             | 67.953 | 64.342 | 56.097 | 62.813 | 19.5710    |
| train best - focal+ts - none (failed) | 23.733 | 5.601  | 2.348  | 10.511 | N/A        |

| mIoU (val)                  | Head     | Common | Tail  | All    | All (test) |
|-----------------------------|----------|--------|-------|--------|------------|
| val best - ce - none        | 38.413   | 7.759  | 3.071 | 16.328 | 17.6254    |
| val best - ce - focal+ts    | 37.370   | 7.327  | 3.048 | 15.829 | 17.7206    |
| val best - focal - none     | 39.909   | 10.169 | 4.227 | 18.022 | 19.3887    |
| val best - focal - focal+ts | 39.280   | 9.621  | 3.904 | 17.521 | 18.7903    |
| none - focal - none         | training |        |       |        |            |

ts = tail sample  
ce = cross entropy


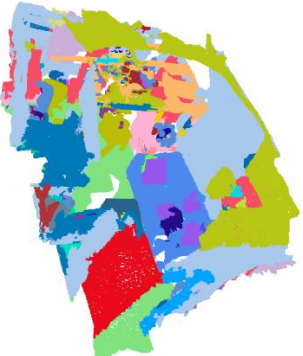

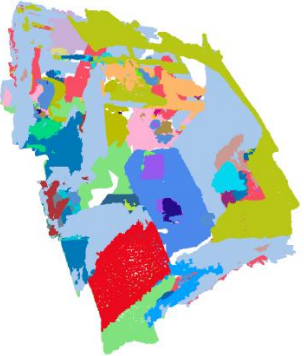


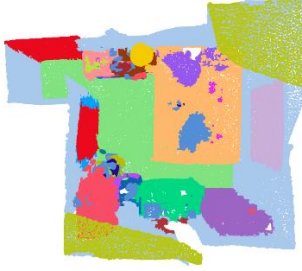









We only have access on train set, and the test set label is unknown. We need to submit to online judge platform to get test set's mIOU score. The first table is training and evaluating on the whole train set, and the second table is we train on partial train set and evaluate on rest of train set (as validation set).

There are three training stage. The original baseline method used two training stage, consisting of: contrastive loss on language grounded label generate by CLIP + weighted cross entropy loss. Here we tried to change the weighted cross entropy loss to focal loss, and adding tail sampling at the last training stage.

We can see from first two rows of table 1, the focal loss improves the performance by 1% mIOU score on test set. The same situation happens in table 2, the model performance improves about 2% mIOU.

However, adding tail sampling does not work as we expect. The model performance was even decreased by 0.5% mIOU on the third and forth rows of table 2. The model even fail to converge in 3<sup>rd</sup> row in table 1.

Visualization result as below:

| Original   | val best – ce - none  | val best – focal - none  | Val best – focal - ts   |
|--|---|--|---|
|    |    |    |    |
|    |    |    |    |
|  |  |  |  |
|  |  |  |  |

## 5 Discussion

We found that input distribution is the key problem in this kind of long-tail problem. The rare class failure largely affects the performance. In the experiment table we can see that, adding focal loss largely increase the model's performance on those tail classes (rare classes). The tail sampling method should have been able to improve tail mIOU too, but it just doesn't. We believe it is because some bugs in our code but we fail to figure it out. The idea of focal loss and tail instance sampling should both tackle down this indoor scene long-tail semantic segmentation problem.

The idea of long-tail 3D scene semantic segmentation is very interesting and quite realistic. While the baseline method proposed by 2022 ECCV paper, [Language-Grounded Indoor 3D Semantic Segmentation in the Wild](#) is quite naïve, there must be a lot of room for improvement. Hope we can further improve it in near future.

## 6 Reference:

[David Rozenberszki, Or Litany, Angela Dai : Language-Grounded Indoor 3D Semantic Segmentation in the Wild. Europe Conference on Computer Vision \(2022\)](#)