Activating Sparse Part Concepts for 3D Class Incremental Learning

Supplementary Material

Table 1. Results on different sizes of the exemplar set.

	500		1000		1500	
	Last	Avg	Last	Avg	Last	Avg
iCaRL	53.70	63.94	61.03	69.36	65.24	72.50
DER	65.13	73.67	69.75	76.72	72.63	78.19
MEMO	61.89	70.55	70.27	77.12	72.17	77.58
Ours	**74.48**	**81.79**	**81.18**	**86.37**	**81.64**	**86.45**

Table 2. Results on different few-shot settings.

	1-shot		5-shot	
	Last	Avg	Last	Avg
iCaRL	3.46	48.82	8.66	19.80
DER	58.55	71.01	59.35	71.90
MEMO	63.22	72.95	53.12	68.87
Ours	**69.98**	**78.40**	**74.36**	**82.80**

1. Hyperparameters Analysis

Our method contains several important hyperparameters, such as the number of part concepts per class N_c, the number of neighbor points in each part K, the number of part sets per shape N_q, and the size of the exemplar set. We show how different choices of these parameters can affect the performance in Fig. 1.

In Fig. 1 (a), we conduct experiments on different numbers of part concepts M for each class, where $M = N_c * C$. When $N_c < 3$, the performance is poor, suggesting that a limited number of part concepts struggle to represent all part patterns effectively. Conversely, performance diminishes as $N_c > 6$, possibly due to the increased difficulty in optimizing a surplus of part concepts.

As shown in Fig. 1 (b), the size of each overlapped part is determined by the number of points it contains, consequently impacting its generalization capabilities. In particular, it is hard for small parts to represent discriminative local geometric patterns. In our experiments, our approach achieves the best results when $K = 64$.

Since we use FPS to sample seeds for part points grouping, a few parts may result in inconsistent part composite features for the same class. As shown in Fig. 1 (c), the results decrease continuously when $N_q < 256$. Our method achieves the best performance when $N_q = 256$.

As depicted in Table 1, we report comparison experiments when using different sizes of the exemplar set $\mathcal{E}_t$. We can observe that our model outperforms all comparison baselines whether we use a large or small exemplar set. Furthermore, our method converges to a stable performance when the size of the exemplar set is greater than 1000. For other methods, we can still observe performance improvements when increasing the exemplar set from 1000 to 1500. A potential reason is that other methods are sensitive to the exemplar set size. This demonstrates the effectiveness of our model when addressing the catastrophic forgetting problem for 3D objects.

2. More analysis

Activated Part Concepts. We highlight the part concept activations in Fig. 2 as learned part concepts according to their similarity w.r.t. point features. We can see that part concepts can activate similar semantic parts of shapes of different classes, such as backs, planes, legs, and screens. This confirms that part concepts can effectively generalize knowledge learned from different classes among different tasks and represent shapes as the composition of part concepts.

Few-shot Settings. We report the results for different few-shot settings in Tab. 2. The size of the exemplar set for each class is set as 1. We observe that our method outperforms other methods under different few-shot settings. It confirms the plasticity of our model to alleviate catastrophic forgetting.

Time and Network Parameters We calculate computation time and network parameters using an NVIDIA Tesla V100 GPU with 32GB memory and an Intel Xeon @2.80GHz CPU with 32 cores. Computation time is measured as the average time of each batch run on Co3D with a batch size of 40 and point clouds with 1024 points. Our method only requires about 8.55M parameters, and each run takes about 0.19s.

Implementation Details. In all experiments, we adopt PointNet as the backbone to extract point-wise features of point clouds sampled 1024 points, therefore $L = 1024$. The Adam optimizer optimizes the network with an initial learning rate of 0.001 and a weight decay of 0.0005. In our experiments, we train the initial task with a batch size of 40 for 200 epochs and the following tasks with the same batch size for 150 epochs. We specify the dimensions of the features and the number of part concepts to 256. The overall method is implemented via PyTorch, and we conduct all experiments on a computer with an NVIDIA Tesla V100 GPU.

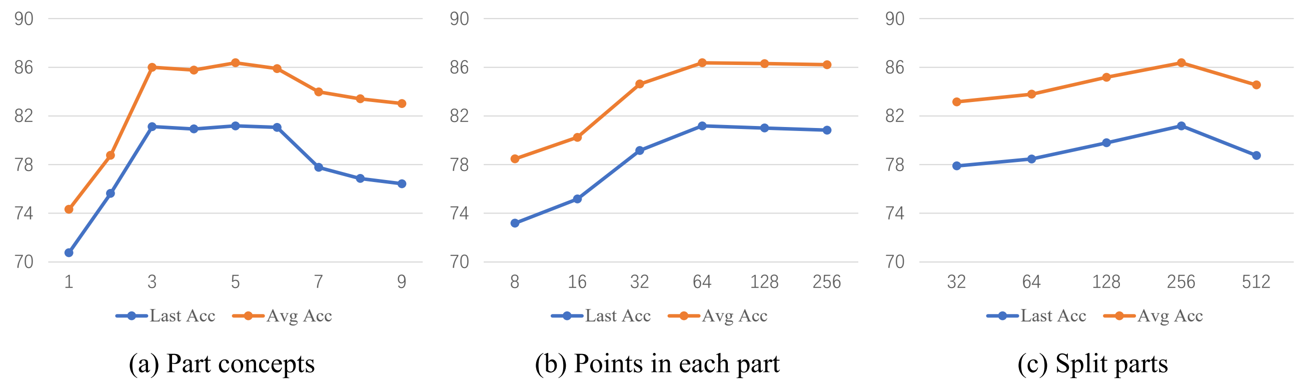

Figure 1. Evaluation of hyperparameters on (a) the number of part concepts for each class, (b) the number of points in each part, and (c) the number of parts for each shape.

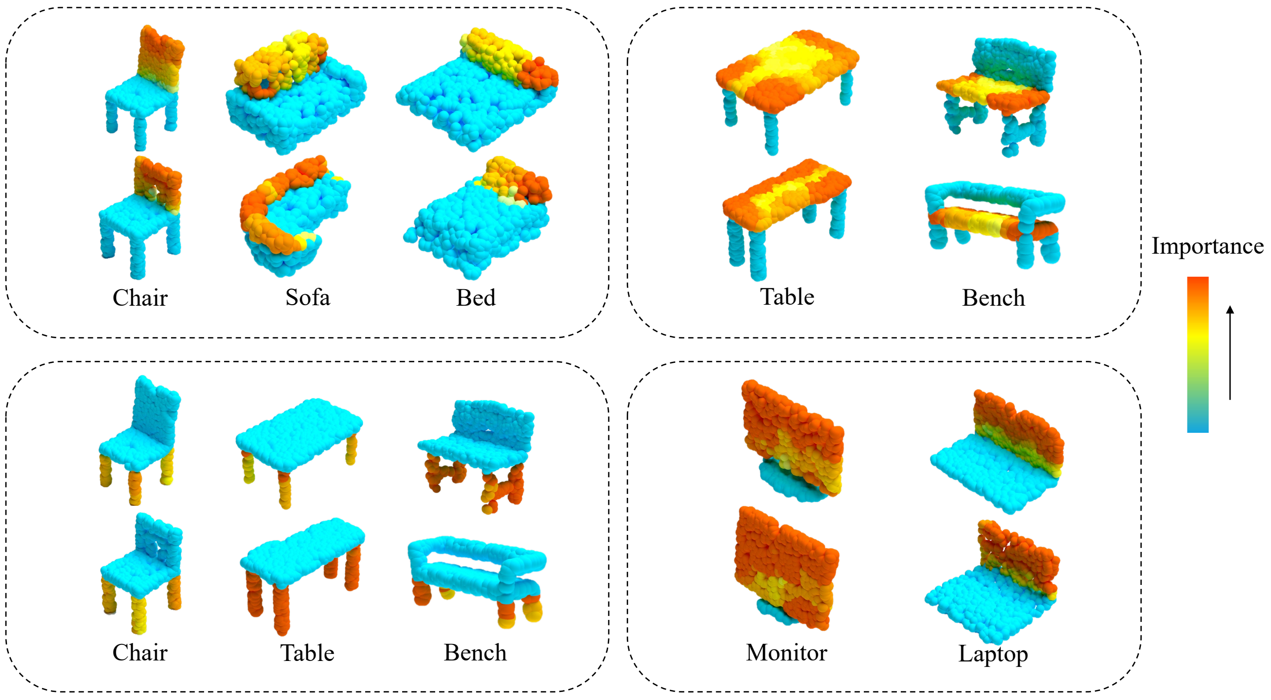

Figure 2. Visualization of different part concepts and their corresponding activations on different shapes.