A. Details for Inter-graph Similarity Computation

Here we present the details for computing the inter-graph similarity through Sikhorn algorithm. Considering a graph $\mathcal{G} = \{F_m, \{D^{m,n}\}_{n=1}^{M}\}_{m=1}^{M}$, we first generate a single-vector representation for each vertex by aggregating other vertices. The aggregation is achieved through weighted sum written as:

$$\hat{F}_m = \frac{1}{\sum_{n=1}^{M} W_{m,n}} \sum_{n=1}^{M} W_{m,n} F_n, \tag{1}$$

Where the weight $W_{m,n}$ is formulated by:

$$W_{m,n} = \exp\left(-D^{m,n}\right). \tag{2}$$

In this way, $\mathcal{G}$ is represented as $\{\hat{F}_m\}_{m=1}^{M}$. For simplify, we denote $\mathcal{G}_i$ for the i-th image as $\{\hat{F}_m^i\}_{m=1}^{M}$. Next, we match $\mathcal{G}_i$ and $\mathcal{G}_j$ by solving an optimal transport (OT) task:

$$\underset{A}{\text{Min}} \sum_{a,b} A_{a,b} M_{a,b}, \tag{3}$$

where A is the transportation plan that implies the alignment information and M is the cost matrix. $M_{a,b}$ measures the transport cost from the a-th vertex $\hat{F}_a^i$ in $\mathcal{G}_i$ to the b-th vertex $\hat{F}_b^j$ in $\mathcal{G}_i$, which is written as:

$$M_{a,b} = 1 - \text{Cos}\left(\hat{F}_a^i, \hat{F}_b^j\right), \tag{4}$$

where Cos denotes the cosine similarity. The unique solution A^* can be calculated through Sinkhorn's algorithm:

$$A^* = \text{diag}\left(\mathbf{u}\right) K \text{diag}\left(\mathbf{v}\right), \tag{5}$$

where the vectors $\mathbf{u}$ and $\mathbf{v}$ are obtained through the above iterations:

$$\begin{aligned} \mathbf{v}^{t=0} &= \frac{\mathbf{1}_m}{\mathbf{v}^{t+1}}, \\ \mathbf{u}^{t+1},\ \mathbf{v}^{t+1} &= \frac{\mathbf{1}_n}{K\mathbf{u}^{t+1}} \end{aligned} \tag{6}$$

we set the iteration number to be 5. Finally, the transport cost tc is computed as:

$$tc = \sum_{a,b} A_{a,b}^* M_{a,b}, \tag{7}$$

which measures the similarity between $\mathcal{G}_i$ and $\mathcal{G}_j$

B. Implementation Details

Our method contains two phases: agent training and policy deployment. The first phase trains the agent network to get the selection policy, while the latter phase employs the trained policy for the CSS training.

For the deployment phase, the hyper-parameters settings follow the previous work [2]. Concretely, we adopt SGD as the optimizer, where the momentum value is 0.9 and the initial learning rate is 1e-2 with the 'poly' learning rate decay schedule. For each continual stage, the network is trained for 30 epochs on Pascal VOC and 60 epochs for ADE20K. The batch size is 24 for both datasets. Following [1], the memory length $|\mathcal{M}|$ is 100 and 300 for Pascal-VOC 2012 and ADE20K, respectively. Following [3], the superpixel number M for computing sample diversity is 5, ϵ in Eq. 5 is 0.1.

For the agent training phase, as we have discussed in Sec. 6 of text, we use the different hyper-parameters settings to speed up training. Concretely, in this phase, we use Deeplabv3 with ResNet18 backbone as the segmentation model. The training epochs Y in Alg. 1 is 1000. We randomly partition 10% of whole data into the training set and leave others as the reward set. For each continual stage, the network is trained for 5 epochs on Pascal VOC and 8 epochs for ADE20K. The segmentation network is optimized by SGD with the initial rate being 0.01, and the agent network is optimized by Monmentumn with the learning rate being 0.1.

C. Segmentation Training

In each stage t of a CSS task, both the memory $\mathcal{M}$ and current dataset $\mathcal{D}_t$ are utilized for training the segmentation model. We follow previous works by using the widely-adopted pseudo-label mechanism to enhance the segmentation training performance. Concretely, the pixels belonging to previous and future classes become the background for images in the current stage. Considering the model is trained on the combined ground truth from both current and previous classes, we use its prediction to generate pseudo labels for background pixels for images in $\mathcal{M}$ and $\mathcal{D}_t$. Concretely, let's denote 0 be the background class. For a sample X with the ground truth label Y, we first use the current segmentation model to get its prediction mask P and the confidence map M, then the pseudo ground truth label $\hat{Y}_i^t$ for the i-th pixel on X is obtained by:

$$\hat{Y}_i^t = \begin{cases} Y_i, & \text{if } Y_i \neq 0 \\ P_i, & \text{if } Y_i = 0 \text{ and } M_i > 0.8 \\ 0, & \text{else} \end{cases} \tag{8}$$

Eventually, X along with the generated pseudo label $\hat{Y}^t$ are used for training the segmentation model through the cross-entropy loss.

D. More Ablation Results

Ablation of Memory Length. In the experiment section, for the fair comparison, we follow [1] by setting

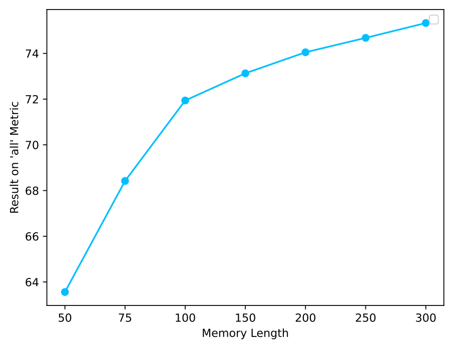

Figure 1. Ablation results of memory length. As the memory length increases from 50 to 300, the mIoU on the 'all' metric increases from 59.10 to 68.37.

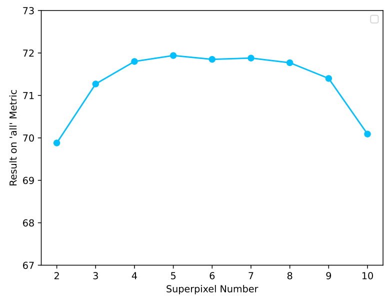

Figure 2. Ablation results of superpixel number.

the memory length $\mathcal{M}$ to 100 and 300 for Pascal-VOC 2012 and ADE20K, respectively. We further validate the performance for the 15-1(6 stages) setting on Pascal-VOC 2012 by using memories with different lengths ranging from 50 to 300. The results are shown in Fig. 1. We can observe that a larger memory brings better performance. As the memory length increases from 50 to 300, the mIoU on the 'all' metric increases from 63.56 to 75.33.

Ablation of Superpixel Number. In order to compute the sample diversity, each region is divided into M superpixels for constructing the graph. Here we perform experimenters to validate how M affects the performance and present the results in Fig. 2. As can be observed, the performance keeps stable when M is larger than 3 and smaller than 9, while a too large M leads to the over segmenting that negatively affects the performance to some extent. Generally speaking, our method is non-sensitive to the hyper-parameter M, demonstrating its high robustness.

E. Discussion of State Representation Computation

As illustrated in Line 435, Sec. 4.2.1 of the text, for computing the sample diversity div and forgetfulness g_c, we introduce a support set $\mathcal{S}_c$ for each class c that contains several graphs for images within c. To relieve the computation burden, for each current class in $\mathcal{C}_t$ that has a larger number of samples, we randomly sample 10% from all images to form $\mathcal{S}_c$. Then sample diversity is derived by computing and averaging the inter-graph similarities with all graphs in $\mathcal{S}_c$. We conduct experiments on the 15-1 (6 stages) setting for Pascal-VOC 2012 dataset to verify the effectiveness. Loading all images into $\mathcal{S}_c$ gets 72.25% mIoU for the 'all' metric, which is just slightly better than the sampled set, which achieves 71.94% mIoU. However, computing similarities on all images consumes 10 times more time than using the sampled set, which is unacceptable. Therefore, our strategy can be computationally efficient yet effective.

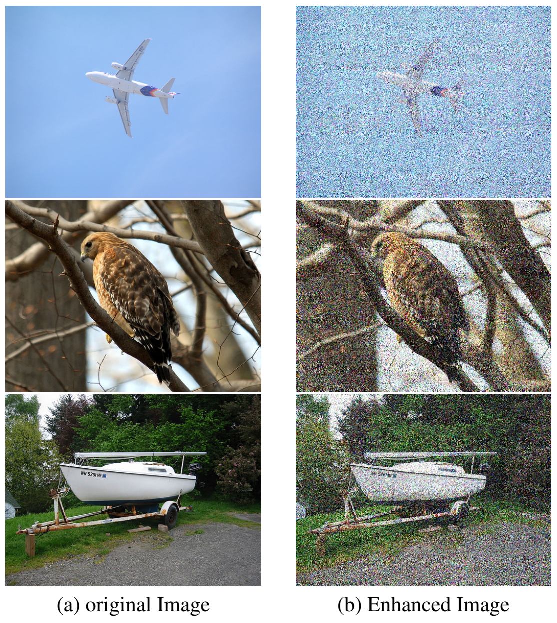

Figure 3. Comparison between the original images and images after enhacement.

F. Cross-dataset Deployment

As discussed in Sec. 6 of the text, we can use an agent trained on one dataset to deploy on other datasets. We perform experiments to verify that capability. For the 'all' metric, using the agent trained on Pascal-VOC 2012 to deploy on the 100-50(2 stages) setting of ADE 20K achieves 34.87% mIoU, and using the agent trained on ADE 20K to deploy on 19-1(2 stages) setting of Pascal-VOC 2012 achieves 74.96% mIoU, with both cases showing good performance. The results demonstrate the high generalization of our method. In realistic applications, the agent only needs to be trained once and then can be used on several different CSS tasks without the extra computation cost for agent retraining.

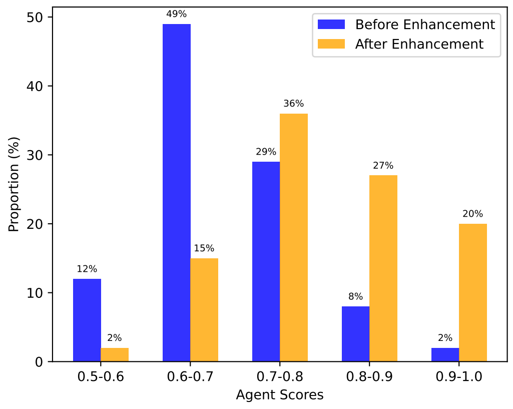

Figure 4. Comparison of agent score distributions for all selected samples before and after enhancement. The horizontal axis represents different score intervals. The vertical axis indicates the proportion of samples falling into each interval.

G. Visualization of Sample Enhancement

Our method includes a novel enhancement action. It enables the selected samples to have the better replay effectiveness by maximizing their agent scores through gradient-based editing. We presents some comparison results between original images and the enhanced images in Fig. 3. We also provide a quantitative comparison in Fig. 4 to show the agent score distributions for all selected samples before and after enhancement, where the horizontal axis represents different score intervals, and the vertical axis indicates the proportion of samples falling into each interval. We can observe that after enhancement, there are more samples with high agent scores. This demonstrates that the gradient-based enhancement effectively increases agent scores, thus promoting the replay performance.

H. Visualization of Segmentation Results

In Fig.5, we present the segmentation results on the Pascal-VOC 2012 validation set using the model trained in the CSS task. We compare our method with the replay approach using the randomly selected samples. Thanks to the proposed mechanism that automatically learns an optimal policy and uses it to select and enhance the most adequate samples, our method can be more effective to alleviate the catastrophic forgetting problem in CSS, thus achieving the better results.

References

[1] Sungmin Cha, YoungJoon Yoo, Taesup Moon, et al. Ssul: Semantic segmentation with unknown label for exemplar-based class-incremental learning. *Advances in Neural Information Processing Systems*, 34:10919–10930, 2021. 1

[2] Arthur Douillard, Yifu Chen, Arnaud Dapogny, and Matthieu Cord. Plop: Learning without forgetting for continual semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4040–4050, 2021. 1

[3] Gen Li, Varun Jampani, Laura Sevilla-Lara, Deqing Sun, Jonghyun Kim, and Joongkyu Kim. Adaptive prototype learning and allocation for few-shot segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8334–8343, 2021. 1

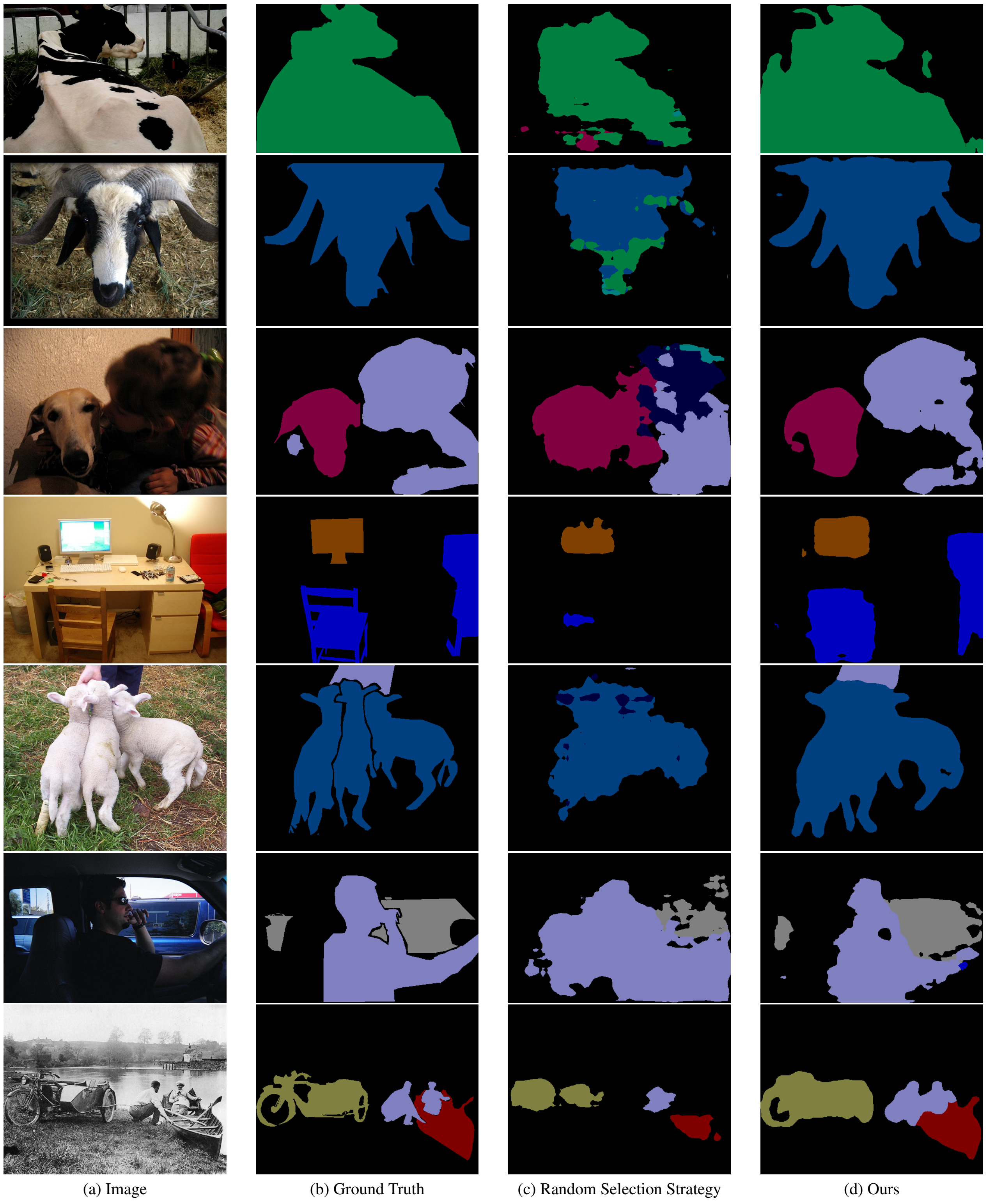

Figure 5. The segmentation visualization comparison results comparison between our method with random selection strategy.