Appendix

Table of Content

A. Additional Results

A.1. Effect of Pseudo-labeling Threshold

As mentioned in Section 3.2 we apply confidence thresholding to remove predicted bounding boxes that have low confidence scores. To show the effectiveness of thresholding, we varied the confidence threshold τ from 0.1 to 0.9 (see Table S1). We observed that the model using a high threshold (*e.g.*, 0.7) yields satisfactory results, as it produces more reliable pseudo-labels with high confidence. On the other hand, using a low threshold can result in lower performance since the model generates too many bounding boxes, which are likely to be false positives.

τ	0.1	0.3	0.5	0.6	0.7	0.8	0.9
FAP (↑)	30.33	31.16	32.17	32.54	33.92	32.81	32.07
CAP (↑)	21.45	21.67	22.38	22.51	23.04	22.69	22.70
F (↓)	-7.11	-7.29	-6.88	-6.98	-7.71	-6.60	-6.94

Table S1. **Ablation study of varying confidence threshold τ on OAK dataset at annotation cost 12.5%.**

A.2. Effect of RPN Loss in Pseudo Training

In Section 3.2, we mentioned that pseudo losses are only applied at the ROI module but not at the RPN module. As shown in Table S2, the model with and without RPN loss in training pseudo-labeled frames show similar performance. We assumed that the RPN module is less likely to suffer catastrophic forgetting since its primary function is to produce general proposals that are class agnostic. As a result, we removed the RPN loss during pseudo training, which also reduces the overall computational cost.

RPN Loss	FAP (↑)	CAP (↑)	F (↓)
✗	**33.92**	**23.04**	**-7.71**
✓	33.64	22.68	-7.62

Table S2. **Effect of RPN loss in pseudo training on OAK dataset at annotation cost 12.5%.**

A.3. Effect of Pseudo Loss Weights

As mentioned in Section 3.2, λ_{pseudo} is a hyperparameter balancing the importance of supervised loss ($\mathcal{L}_{sup}$) and pseudo loss ($\mathcal{L}_{pseudo}$). To examine the effect of λ_{pseudo}, we varied the λ_{pseudo} from 0.5 to 4.0 at annotation cost 12.5% on OAK dataset. As shown in Table S3, the model performs the best with $\lambda_{pseudo} = 1.0$ and shows moderate performance drop for other values of λ_{pseudo} (0.5, 1.5, and 2.0). However, when λ_{pseudo} is set to 4.0, the model performance deteriorates.

λ_{pseudo}	0.5	1.0	1.5	2.0	4.0
FAP (↑)	33.19	33.92	33.01	32.67	30.08
CAP (↑)	22.92	23.04	22.52	22.02	20.17
F (↓)	-6.55	-7.71	-7.71	-7.86	-7.50

Table S3. **Ablation study of varying pseudo loss weights λ_{pseudo} on OAK dataset at annotation cost 12.5%.**

A.4. Effect of Data Augmentation

As mentioned in 3.2, we use data augmentation techniques when training the pseudo-labeled frames. Here we ablated our Efficient-CLS by removing data augmentation in pseudo training. From Table S4 at annotation cost 12.5%, we observed that removing data augmentation in pseudo training leads to a performance drop of 3.60% in FAP, 1.96% in CAP and 1.21% in F on OAK dataset. This indicates that using data augmentation on pseudo-labeled frames can enforce the model to learn invariant object representations from these video frames.

Data Augmentation	FAP (↑)	CAP (↑)	F (↓)
✗	30.32	21.08	-6.50
✓	**33.92**	**23.04**	**-7.71**

Table S4. **Effect of data augmentation in pseudo training on OAK dataset at annotation cost 12.5%.**

A.5. Design Decisions on External Replay Buffer

First, **we explored whether a replay buffer needs to be balanced based on class distribution in the video streams**. Our current design ensures that there are at least 5 images containing object instances of any given learnt classes. In comparison, we conducted an additional experiment (Random Store and Replay) where we designed a replay buffer of the same size as Efficient-CLS and saved any new images regardless of the class labels. Ideally, this replay buffer represents the imbalanced class distribution of the video streams. For example, cars appear more often than stop signs; thus, it is more likely to store more images containing cars in the replay buffer. In Table S5, we found that a class balanced replay buffer performs much better than Random Store and Replay, implying the importance of class balanced replay buffer.

	FAP (↑)	CAP (↑)	F (↓)
Random Store and Replay	26.23	21.45	-3.23
Balanced Store and Replay	**40.24**	**28.18**	**-8.10**

Table S5. **Effect of class balanced replay buffer on OAK dataset in fully supervised protocol.**

Second, **we studied how many frames are needed to retrieve from the external buffer for replay** in conjunction with the batch of video frames in the current training iteration (Table S6). To perform episodic replay, we randomly retrieve 16 video frames from the replay buffer for joint training with current frames of a mini-batch. As shown in Table S6, replaying 16 video frames is a good trade-off between model performance and extra training time. Replaying fewer samples would lead to poor performance and forgetting, while replaying more hardly brings any benefits but largely increases the training resources. We compared with the implementation of Wang et al. [34] where the batch size for replays increases according to the number of seen classes in each iteration. Table 1 demonstrates the effectiveness of our replay technique (iCaRL(our impl.) *vs.* iCaRL(Wang et al.)).

	Replay Size	FAP (↑)	CAP (↑)	F (↓)
iCaRL (our impl.)	2	14.14	10.62	-0.09
	4	21.93	15.26	-0.57
	8	34.08	22.20	-4.83
	16	**36.14**	**26.26**	**-4.89**
	32	37.42	28.28	-3.09
	64	35.59	27.81	-1.65
Efficient-CLS	2	18.79	13.37	-2.05
	4	25.52	18.22	-4.64
	8	37.03	24.32	-8.21
	16	**40.24**	**28.18**	**-8.10**
	32	41.04	30.01	-7.90
	64	38.79	29.92	-7.61

Table S6. **Ablation study of the number of replay sample per training step on OAK dataset in fully supervised protocol.** The buffer size is set to 5 images per class. Our choices are **bold-faced**.

Third, **we studied the effect of the number of sample images stored per class in the replay buffer**. We varied the number of sample images saved per class. As the number of sample images per class increases, it increases the diversity of the object representations per class; hence leads to steady performance boost (Table S7). This trend is observed in both iCaRL and Efficient-CLS. Of course, one could argue that the ideal case is to store all the past video frames and replay all of them. This reverts to the offline setting and yields the best model performance in LEOCOD. However, it is at the expense of heavy usage of memory storage. In practice, one has to strike a nice balance between model performance and memory storage. Here, we demonstrate that even saving 5 images per class, Efficient-CLS surpasses all the competitive baselines in LEOCOD.

	Buffer Size	FAP (↑)	CAP (↑)	F (↓)
iCaRL (our impl.)	1	28.60	20.60	-2.59
	5	**36.14**	**26.26**	**-4.89**
	10	39.96	28.37	-6.45
	15	40.56	28.84	-6.69
	20	41.26	29.16	-7.19
	30	42.04	29.57	-7.57
	50	43.22	30.21	-7.53
Efficient-CLS	1	31.41	22.50	-6.74
	5	**40.24**	**28.18**	**-8.10**
	10	43.44	29.96	-8.80
	15	44.57	30.17	-8.98
	20	45.10	30.36	-8.46
	30	45.26	30.87	-9.37
	50	46.95	31.33	-9.00

Table S7. **Ablation study of varying numbers of samples per class stored in replay buffer on OAK dataset in fully supervised protocol.** The replay size is set to 16 images per training iteration. Our choices are **bold-faced**.

A.6. Analysis of Unlabeled Frames Selection

We conducted Efficient-CLS with 5 runs. Each run applies a different random seed for unlabeled frames selection. We reported the means and standard deviations of these 5 runs in Table S8. We found that our Efficient-CLS shows reliable and robust performance against different selections of unlabeled frames in the video stream.

Annotation Cost (%)	50	25	12.5	6.25
FAP (↑)	38.45 (±0.68)	38.00 (±1.17)	34.29 (±0.76)	30.50 (±1.07)
CAP (↑)	26.85 (±0.24)	26.38 (±0.32)	23.47 (±0.73)	20.60 (±0.43)
F (↓)	-8.01 (±0.77)	-8.32 (±0.98)	-7.30 (±0.82)	-6.28 (±0.69)

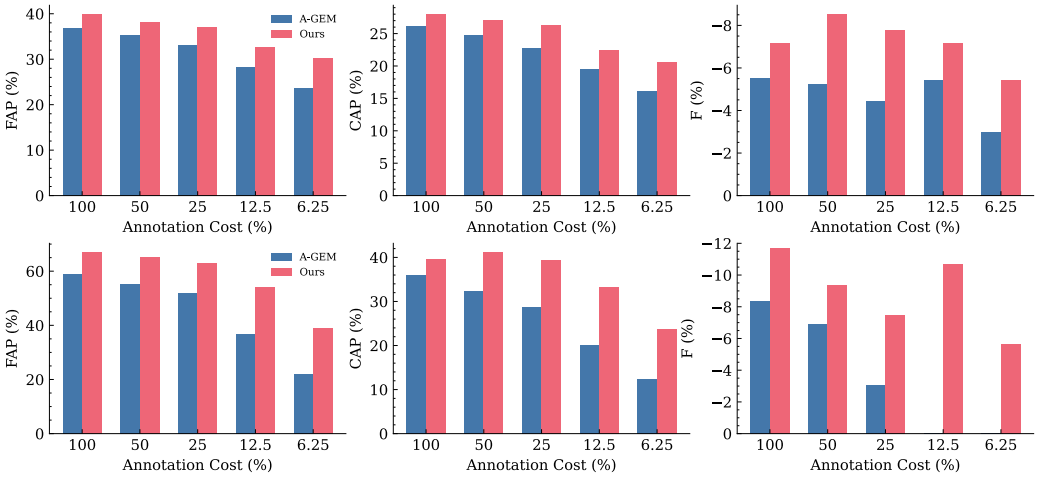
Table S8. **Performance of our Efficient-CLS on OAK dataset in sparse annotation protocol.** The table header denotes the percentage of frames that are labeled in the video stream. The means and standard deviations in brackets are reported.

A.7. Visualization of Pseudo-labeling

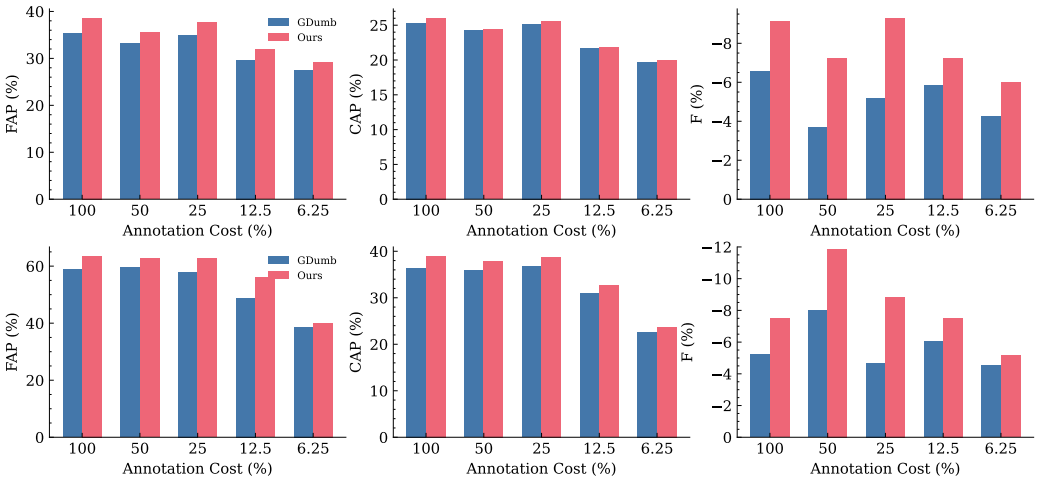
As shown in Figure S2, the pseudo-labels generated by our Efficient-CLS (2nd column) capture more ground truth objects and contain fewer false positive instances than the Naive Pseudo-labeling model (1st column).

B. Limitations and Future Work.

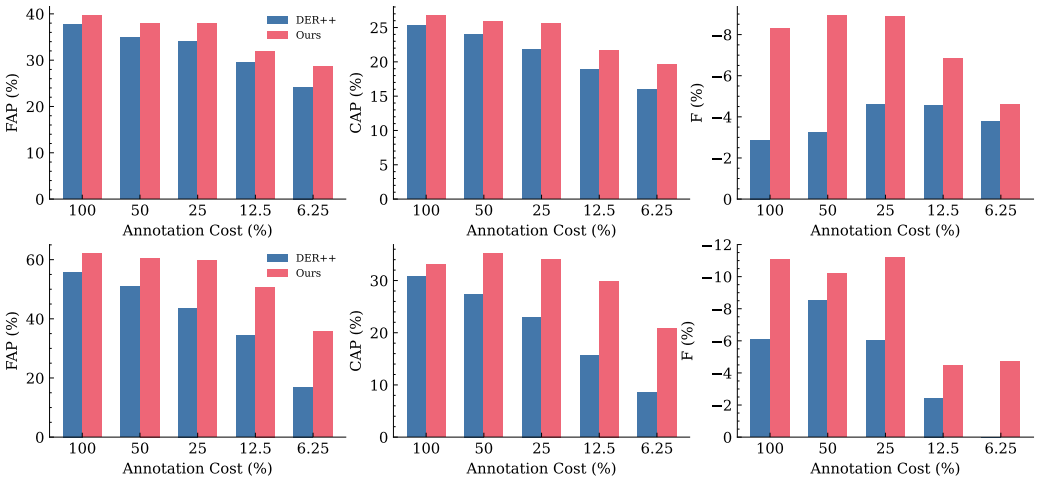
Same as other replay methods, our method is facing with infinite memory expansion problem. Our replay buffer stores 5 images per object class. As the number of seen classes increases, the memory buffer has to expand. One could imagine that this will be a challenging case in lifelong learning on video streams which last for tens of years and thousands of classes have to be learnt. In the future, we will explore more efficient replay strategies with latent object representations.



A) Comparisons of A-GEM (blue) and A-GEM *w/* Efficient-CLS (red).



B) Comparisons of GDumb (blue) and GDumb *w/* Efficient-CLS (red).



C) Comparisons of DER++ (blue) and DER++ *w/* Efficient-CLS (red).

Figure S1. **Evaluation of state-of-the-art CL methods in LEOCOD setting on OAK dataset (first row) and EgoObjects dataset (second row)**. The higher the bars are, the better. The x-axis denotes the percentage of video frames that are labeled in the video stream. It ranges from 6.25% to 100% (full supervision). The y-axis indcates the performance using different evaluation metrics.

Figure S2. **Visualization of example pseudo-labels predicted by our Efficient-CLS and the Naive Pseudo-labeling.** The white box with dash line denotes the ground truth label. The box with solid line denotes the pseudo-labels (the ones in green are correct while the red are wrong labels). The **Naive Pseudo-labeling** only has one learner and uses the pseudo-labels generated by itself for training.