LABEL-EFFICIENT ONLINE CONTINUAL OBJECT DETECTION IN STREAMING VIDEO

Anonymous authors
Paper under double-blind review

ABSTRACT

To thrive in evolving environments, humans are capable of continual acquisition and transfer of new knowledge, from a continuous video stream, with minimal supervision, while retaining previously learnt experiences. In contrast to human learning, most standard continual learning (CL) benchmarks focus on learning from static i.i.d. images that all have labels for training. Here, we examine a more realistic and challenging problem—Label-Efficient Online Continual Object Detection (LEOCOD) in streaming video. By addressing this problem, it would greatly benefit many real-world applications (e.g., personalized robots, augmented/virtual reality headsets, etc.) with reduced data annotation costs and model retraining time. To tackle this problem, we seek inspirations from complementary learning systems (CLS) in human brains and propose Efficient-CLS, a plug-and-play module that can be easily inserted into and improve existing continual learners. On two challenging CL benchmarks for streaming real-world videos, we integrate Efficient-CLS into state-of-the-art CL algorithms, and achieve significant improvement with minimal forgetting across all supervision levels. Remarkably, with only 25% annotated video frames, our Efficient-CLS still outperforms the base CL learners, which are trained with 100% annotations on all video frames. We will make source code publicly available upon publication.

1 INTRODUCTION

Humans have the ability to continuously learn from an ever-changing environment, while retaining previously learnt experiences. In contrast to human learning, prior works (Aljundi et al., 2019b;a; Fini et al., 2020; Wang et al., 2021a; Caccia et al., 2022) show that deep neural networks (DNNs) are prone to catastrophic forgetting. To address the forgetting problem, existing works in continual learning (CL) primarily focus on class-incremental image classification or object detection. Their experiment settings are often idealistic and simplified, where **i.i.d. static images** are usually grouped by class and incrementally presented to computational models in sequence. To learn a particular task containing specific classes, an agent can go through the entire dataset of current task (not the previous ones) **over multiple epochs**. After that, the learned classes in current task become unavailable, *i.e.* **no overlaps** between the sets of learned classes and unseen classes.

However, these experiment designs deviate from the online continual learning (OCL) setting in the real world, where an agent learns from **temporally correlated non-i.i.d video streams** in **one single pass**. Given context regularities in natural environments, an agent is likely to encounter cases when objects of previously learnt classes **co-occur** with unknown objects from unseen classes, *e.g.* a computer mouse and a computer monitor often co-occur. Taking these considerations, Wang et al. (2021a) introduces OCL on object detection in real-world video streams. They evaluate existing CL approaches on this setting and report a huge performance gap compared with offline training.

Based on the setting in Wang et al. (2021a), we take a significant step further and introduce a novel problem setting called Label-Efficient Online Continual Object Detection (LEOCOD), which highlights another two unique challenges. **First**, the setting in Wang et al. (2021a) is designed in a way that computational models are trained with every mini-batch over multiple passes. We tighten the training recipe in LEOCOD to strictly online, where data is allowed to have one single pass and models are trained on the entire video dataset for only one epoch. **Second**, the existing CL models require fully supervised training where box-level ground truth labels of every object on every video

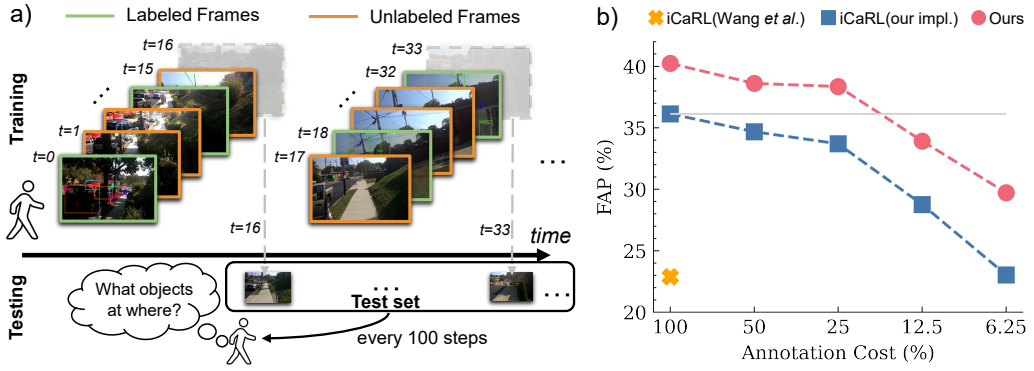


Figure 1: **(a) Problem introduction**: An agent continuously learns from a never-ending online video stream over time. In each training step, out of a mini-batch containing 16 consecutive video frames, only a fixed proportion of frames are labeled (green boundary), while the rest of the frames are unlabeled (orange boundary). Following Wang et al. (2021a), the video frame after every training mini-batch (transparent) is held out for testing. After every 100 training steps, the agent is evaluated on all the video frames from the test set for object detection. **(b)** Our proposed method (red) consistently outperforms the best competitive baseline (blue) by a margin of 5%. Remarkably, our model, trained at 25% annotation cost, surpasses the best baseline trained at 100% (grey line). The orange cross denotes the performance of the state-of-the-art model, which is 15% lower than ours.

frame have to be obtained from human annotators. Unlike static images, acquiring human annotations for object detection on videos can be expensive and daunting. Thus, in LEOCOD, the video frames per mini-batch are sparsely annotated to alleviate the burdens of real-time human labeling. This makes LEOCOD more feasible for online practices in the real world (see Appendix B.6 for feasibility analysis). The reduce of training iterations and supervision lead to inferior performance in the existing CL models, which we aim to address.

Cognitive science works (Wang et al., 2020; Lake et al., 2017) show that humans are efficient at continuously learning from very few annotated data samples. We get inspirations from the theory of Complementary Learning Systems (CLS) in human brains (Kumaran et al., 2016), and propose a plug-and-play module for the LEOCOD task, dubbed as Efficient-CLS. In Efficient-CLS, we introduce two feed-forward neural networks as slow and fast learners. In the fast learner, memory is rapidly adapted to the current task. The weights of the slow learner change a little on each reinstatement, and are maintained by taking the exponential moving average (EMA) of the fast learner's weights over time. Though a few continual learning models in previous works (Arani et al., 2022; Pham et al., 2020) also use a similar source of inspiration, they miss the effect of reciprocal connections from slow learners to fast learners, which we intend to address. Inspired by the bidirectional interaction in CLS (Ji & Wilson, 2007), we reactivate the weights of the slow learners to predict meaningful pseudo labels from the unlabeled video frames and use these pseudo labels to guide the training of the fast learner, closing the loop between the two systems. We provide detailed discussions on neuroscience inspiration in Applendix B.7.

We demonstrate the *versatility* and *effectiveness* of our Efficient-CLS on two standard real-world video datasets, OAK (Wang et al., 2021a) and EgoObjects[1]. Our proposed Efficient-CLS can be easily integrated into existing CL models and consistently improve their performance by a large margin in LEOCOD. It is worth noting that, with only 25% labeled data, Efficient-CLS surpasses the comparative baselines trained with full supervision (Figure 1(b)).

Our contributions of this paper are two-fold:

- We introduce a new, challenging and important problem of label-efficient online continual object detection (LEOCOD) in video streams. Solving this problem would greatly benefit real-world applications in minimizing annotation cost and reducing model retraining time.
- To tackle this problem, we propose Efficient-CLS, a plug-and-play module inspired from the theory of Complementary Learning Systems. It can be integrated into existing CL models and learn efficiently and effectively with less supervision and minimal forgetting.

[1]https://sites.google.com/view/clvision2022/challenge

2 Related Work

Continual Learning (CL). To alleviate catastrophic forgetting, many CL approaches follow the standard of maintaining an external buffer where a limited number of old samples are stored and used for replay when adapting to a new task. iCaRL (Rebuffi et al., 2017) stores the representative exemplars in past tasks for knowledge distillation and prototype rehearsal. Gradient Episodic Memory (GEM) (Lopez-Paz & Ranzato, 2017) formulates optimization constraints on the exemplars in memory. Averaged GEM (A-GEM) (Chaudhry et al., 2018) is an improved version of GEM that achieves faster training and less memory consumption. GDumb (Prabhu et al., 2020) greedily stores samples in memory as they come and trains a model using samples only in the memory. Dark Experience Replay++ (DER++) (Buzzega et al., 2020) combines replay with knowledge distillation and regularization, and samples logits along the entire optimization trajectory.

Although the disjoint task formulation places a constrain on the data space. It does not put any restrictions on the learner itself, *i.e*. the learner can perform multiple passes over entire dataset corresponding to current task. In contrast, the online continual learning (OCL) paradigm tightens this constrain on the learner to allow only one single pass over the training data. This reduces computational costs while makes the CL setting more challenging. Lately, Wang et al. (2021a) benchmark OCL for object detection in real-world streaming video. They identify a large performace gap between existing CL methods and offline training, which we aim to address. Besides, as the video streams arrive endlessly in a real-time manner, assigning annotations to all the video frames for training is laborious and time-consuming. It becomes even more daunting in object detection tasks where class labels and bounding boxes of all objects on a video frame have to be provided.

To involve less human labeling, we propose a novel setting called label-efficient online continual object detection (LEOCOD), where only a small proportion of video frames per mini-batch are labeled while the rest remain unlabeled (see Figure 1). This design shares similar motivations with the latest works in semi-supervised continual learning (SSCL) (Smith et al., 2021; Wang et al., 2021b; Boschini et al., 2022). However, their settings are still based on offline CL which allows unrestricted access to data that belongs to the same task. Moreover, their methods rely heavily on the task boundaries that provide additional task information for image classification (*e.g*., the change of learning classes), which are not available for object detection in online streaming video. Instead, our proposed Efficient-CLS is task-free. It can be easily plugged into existing CL models and play an essential role in reducing annotation costs and catastrophic forgetting.

Complementary Learning Systems (CLS). The essence of fast and slow learning in CLS has benefited several continual learning applications in image recognition (Pham et al., 2020; 2021; Rostami et al., 2019; Arani et al., 2022; Kamra et al., 2017). However, these methods either require the task boundaries, which are not applicable in our online video setting, or they require to train fast and slow learning systems with replay samples from the same replay buffer, which could easily lead to overfitting problem when the replay buffer has limited capacity. To eliminate overfitting problem, Rostami et al. (2019) and Kamra et al. (2017) utilize generative replay models to couple sequential tasks in a latent embedding space. While generative approaches have succeeded in artificial and simple datasets, they often fail in complex vision tasks, *e.g*., object detection. Based on the neuroscience evidence of the bidirectional interaction between the hippocampus and the neocortex (Dudek & Bear, 1993), we leverage slow learners to exploit unlabeled video frames and generate pseudo labels for training fast learners. The semantic replays via pseudo-labeling encourage fast learners to capture more generic representations from diverse data of unlabeled video frames; hence, in turn, contributing to reinstatement of memory in slow learners, resulting in a positive feedback loop.

3 Label-Efficient Online Continual Object Detection

3.1 Problem Setting

We advance the online continual object detection in Wang et al. (2021a) to a **label-efficient** and **computationally-efficient** setting—Label-Efficient Online Continual Object Detection (LEOCOD). In contrast to the setting in Wang et al. (2021a) where video frames are *extensively annotated* and trained with *multiple epochs*, an agent in LEOCOD continuously learns from a *sparsely annotated* video stream in *a single pass* over time (see Figure 1(a)).

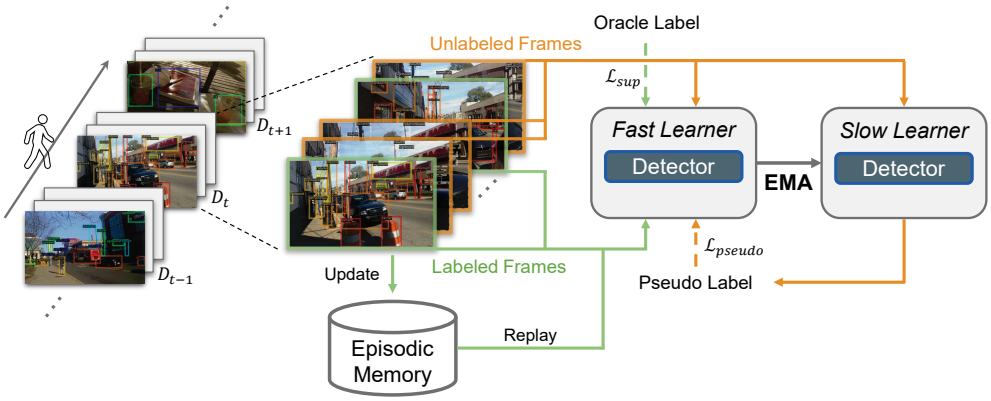


Figure 2: **The overview of Efficient-CLS**. At each learning step, the system receives a batch of temporally continuous data D_t, including labeled (green) and unlabeled (orange) frames. The fast learner trains the labeled frames alongside a small subset of labeled exemplars retrieved from episodic memory with the supervised loss $\mathcal{L}_{sup}$. Meanwhile, the fast learner leverages the pseudo labels generated by the slow learner to optimize a pseudo loss $\mathcal{L}_{pseudo}$. To reinstate memory of the slow learner, the synaptic weights of the slow learner are updated by taking the Exponential Moving Average (EMA) of the fast learner's weights. The fast and slow learners are complementary to each other, forming a positive feedback loop.

Formally, we consider the online continual object detection on a continuum of video streams $\mathcal{D} = \{D_1, \cdots, D_T\}$ where at time step t, a learning agent receives a mini-batch of continuous video frames D_t from current environment for online training (one single pass). To perform label-efficient object detection, within the batch D_t, only a subset of video frames $D_t^s = (X_t^s, Y_t^s)$ are labeled, while the remaining video frames $D_t^u = (X_t^u)$ are unlabeled. For each labeled data sample, its annotation contains the bounding box locations and their corresponding class labels.

3.2 Efficient-CLS: Efficient Complementary Learning Systems

We propose a plug-and-play module dubbed as Efficient-CLS. Specifically, it consists of two feed-forward networks: (i) the fast learner is designed to quickly encode new knowledge from current data stream and then consolidate it to the slow learner; and (ii) the slow learner accumulates the acquired knowledge from fast learner over time and guides the fast learner with meaningful pseudo labels, when full supervision is not available. Following Rebuffi et al. (2017), we maintain an external episodic memory, as a replay buffer, to store exemplars that can be retrieved for replays alongside ongoing video stream. As the fast and slow learners are model agnostic, our Efficient-CLS can be easily integrated into existing CL models, which leads to less supervision and minimal forgetting.

3.2.1 Learning with Labeled Frames

The fast learner and slow learner use the same standard Faster-RCNN (Ren et al., 2015) detector f. Despite the same architecture, the weights of the fast and slow learners are not shared. We use θ_F and θ_S to denote the network parameters for fast and slow learners respectively. As shown in Figure 2, at each training step t, we use the labeled video frames $D_t^s = (X_t^s, Y_t^s)$ to optimize the fast learner θ_F with the standard supervised loss $\mathcal{L}_{sup}$ in Faster-RCNN (Ren et al., 2015). It consists of four losses: Region Proposal Network (RPN) classification loss $\mathcal{L}_{cls}^{rpn}$, RPN regression loss $\mathcal{L}_{reg}^{rpn}$, Region of Interest (ROI) classification loss $\mathcal{L}_{cls}^{roi}$, and ROI regression loss $\mathcal{L}_{reg}^{roi}$. We define $\mathcal{L}_{sup}$ as:

$$\mathcal{L}_{sup} = \mathcal{L}_{cls}^{rpn}(X_t^s, Y_t^s) + \mathcal{L}_{reg}^{rpn}(X_t^s, Y_t^s) + \mathcal{L}_{cls}^{roi}(X_t^s, Y_t^s) + \mathcal{L}_{reg}^{roi}(X_t^s, Y_t^s). \tag{1}$$

3.2.2 Learning with Unlabeled Frames

We introduce a pseudo-labeling paradigm to capitalize the information from unlabeled video frames $D_t^u = (X_t^u)$ for training. In our early exploration, we intuitively use the fast learner for pseudo-labeling as it quickly adapts the knowledge of nearby frames. However, we observe that using

the pseudo labels generated by the fast learner for self-replay exhibits biases towards recently seen objects, which is less effective in preventing forgetting. This has also been verified in our ablation study (Section 5.3). In contrast, the slow learner preserves the semantic knowledge over a longer time span which generates pseudo labels with fewer biases. This encourages the fast learner to capture more generic scene representations, hence, in turn, contributing to reinstatement of memory in the slow learner (Section 3.2.3), resulting in a positive feedback loop.

Given all these design considerations, the slow learner takes the unlabeled video frames D_t^u as inputs to estimate the possible objects of interest and their corresponding bounding box locations. For brevity, we refer these "pseudo bounding boxes and their corresponding class labels" as "pseudo labels" in the paper. To get rid of false positives, we apply a threshold τ to filter out bounding boxes with predicted low confidence scores. Moreover, there also exist repetitive boxes which negatively impact the quality of pseudo-labeling. To address this issue, we use the technique of class-wise non-maximum suppression (NMS) (Ren et al., 2015) to remove the overlapped boxes and get the high-quality pseudo labels. Formally, the procedure of pseudo label generation is summarized below:

$$Y_t^u = NMS([f(X_t^u; \theta_S)]_{>\tau}), \tag{2}$$

where $[\cdot]_{>\tau}$ denotes the bounding box selection with confidence score larger than τ.

Given that the video streams are captured from the egocentric perspective in the real world, head and body motions may lead to undesired motion blur effects on some video frames. To enforce our module to learn invariant object representations from these video frames, same as the previous work (Zoph et al., 2020), we apply data augmentation techniques on the pseudo-labeled frames, including 2D image crops, rotations, and flipping. Note that different from image classification, the predicted bounding box locations also need to be updated accordingly after image augmentations. We denote these pseudo-labeled video frames and their re-adjusted pseudo labels after data augmentations as $(\tilde{X}_t^u, \tilde{Y}_t^u)$. We can then use these pseudo pairs $(\tilde{X}_t^u, \tilde{Y}_t^u)$ to train the fast learner by optimizing the pseudo loss $\mathcal{L}_{pseudo} := \mathcal{L}_{cls}^{roi}(\tilde{X}_t^u, \tilde{Y}_t^u) + \mathcal{L}_{reg}^{roi}(\tilde{X}_t^u, \tilde{Y}_t^u)$. Note that we only apply pseudo losses at the ROI module, as we empirically verified that the RPN module has no effects on pseudo training (see Appendix C.4).

Overall, our Efficient-CLS is jointly trained with the following losses: $\mathcal{L}_{total} = \mathcal{L}_{sup} + \lambda_{pseudo}\mathcal{L}_{pseudo}$, where λ_{pseudo} is the weight of $\mathcal{L}_{pseudo}$.

3.2.3 Synapses Consolidation via Exponential Moving Average

To alleviate forgetting of obtained knowledge, we apply Exponential Moving Average (EMA) to gradually update the slow learner with the fast learner's synaptic weights. The evolving synaptic changes in the slow learner are functionally correlated with the memory consolidation mechanism in the hippocampus and the neocortex (Arani et al., 2022). Formally, we define EMA process as:

$$\theta_S = \alpha\theta_S + (1 - \alpha)\theta_F, \tag{3}$$

where the $\alpha \in [0, 1]$ is EMA rate. According to the stability-plasticity dilemma, a smaller α means faster adaption but less memorization. Empirically, we set $\alpha = 0.99$, which leads to best performance (see Appendix C.2 for detailed analysis on choices of α).

4 Experimental Details

4.1 Datasets

We consider two challenging datasets, *i.e.*, OAK (Wang et al., 2021a) and EgoObjects[2] for online continual object detection on video streams.

OAK dataset is a large egocentric video stream dataset spanning nine months of a graduate student's life, consisting of 7.6 million frames of 460 video clips with a total length of 70.2 hours. The dataset contains 103 object categories. We follow Wang et al. (2021a) in the ordering of training and testing data splits. One frame every 16 consecutive video frames lasting for 30 seconds is held out to construct a test set and the remaining frames are used for training.

[2]https://sites.google.com/view/clvision2022/challenge

EgoObjects is one of the largest object-centric datasets focusing on object detection task. It includes 40,000 videos (around 110 hours), covering 600 object categories. We take a subset of EgoObjects to benchmark LEOCOD (see Appendix A.4 for details). For consistency, we use the same ordering above as OAK dataset to construct the train and test data splits.

4.2 BASELINES

We compare our model against the following baselines: 1) *Vanilla training*: **Incremental** is a naive baseline trained sequentially over the entire video stream without any measures to avoid catastrophic forgetting; **Offline Training** is an upper bound which trains the entire data stream over multiple epochs. 2) *State-of-the-art CL algorithms*: **EWC** (Kirkpatrick et al., 2017), **iCaRL** (Rebuffi et al., 2017), **A-GEM** (Chaudhry et al., 2018), **GDumb** (Prabhu et al., 2020), **DER++** (Buzzega et al., 2020) and **iOD** (Kj et al., 2021).

The iCaRL model implemented by Wang et al. (2021a) stands as the state-of-the-art (SOTA) method in online continual object detection. We reproduce their results using the released code[3]. When calculating RPN and ROI losses for replay samples, their iCaRL model neglects the losses of background proposals and penalizes the foreground losses according to the proportion of the current samples and replay samples. We empirically find that this trick hinders the model from effective episodic replay, thus resulting in severe forgetting. Therefore, we re-implement the iCaRL by discarding the re-weighting trick and reverting back to the standard RPN and ROI losses. We name these two different implementations as **iCaRL(Wang *et al.*)** and **iCaRL(our impl.)**, respectively.

4.3 EVALUATION

Protocols. First, we define the annotation cost as the proportion of number of labeled frames versus the total 16 frames within a mini-batch D_t. For example, if 2 out of 16 consecutive frames within D_t get labeled, the annotation cost is $2/16 = 12.5\%$. The frames to be labeled are randomly selected within each mini-batch D_t. Considering that different choices of labeled frames might influence the model performance, for fair comparisons between models, we fix the choice of randomly selected labeled frames and use the same labeled and unlabeled frames for training all models. We found that our Efficient-CLS shows reliable and robust performance against different selections of unlabeled frames in the video stream (see Appendix B.2). Based on the various annotation costs, we introduce two training protocols: *fully supervised protocol* (100% annotation cost) and *sparse annotation protocol* (where the annotation cost is less than 100%). In sparse annotation protocol, we further split the training experiments based on 50%/25%/12.5%/6.25% annotation costs.

Testing. We use the same test set for evaluating computational models. As shown in Figure 1, we always add the last video frame out of every 16 video frames within a mini-batch D_t to our test set. Once the test set is constructed for each dataset, it is fixed. All the frames in the test set are repetitively used for evaluating computational models at every 100 learning steps.

Metrics. We evaluate these baselines on OAK and EgoObjects datasets with three standard metrics: continual average precision (CAP), final average precision (FAP) and Forgetfulness (F) (Wang et al., 2021a). CAP shows the average performance of a continual learning algorithm over the time span of the entire video stream, while FAP denotes the final performance of a model after seeing the entire video stream. F estimates the forgetfulness of the model due to the sequential training. It takes into account the time interval between the first presence of an object category and its subsequent presence. See Appendix A.3 for their detailed definitions.

4.4 IMPLEMENTATION DETAILS

For fair comparisons, same as Wang et al. (2021a), we use pre-trained Faster-RCNN (Ren et al., 2015) with ResNet-50 backbone (He et al., 2016) on PASCAL VOC (Everingham et al., 2015) for all the continual learning algorithms. For replay-based methods, the replay buffer stores total 5 samples per class (around 500 frames for OAK, and 1400 for EgoObjects). We also fix the number of replay samples to 16 frames per time step, which requires less training time compared with Wang et al. (2021a). For our Efficient-CLS, we use the output of the slow learner at the inference stage, as it excels at avoiding catastrophic forgetting (see Section 5.3). More training and implementation details can be found in Appendix A.1.

[3] https://github.com/oakdata/benchmark

Table 1: **Performance of Efficient-CLS and other state-of-the-art methods on OAK and EgoObjects**. iCaRL(Wang *et al*.) denotes the SOTA model presented in Wang et al. (2021a), and iCaRL(our impl.) is the same method by our implementation.

		OAK			EgoObjects		
	Annotation Cost	FAP (↑)	CAP (↑)	F (↓)	FAP (↑)	CAP (↑)	F (↓)
Incremental	100%	8.38	7.72	0.03	10.21	3.55	1.48
Offline Training	100%	48.28	35.23	-	86.18	59.81	-
EWC	100%	7.73	7.02	-0.12	5.15	1.60	0.57
iOD	100%	7.92	7.14	0.98	8.80	2.64	0.00
iCaRL(Wang *et al*.)	100%	22.89	16.60	-2.95	37.61	21.71	2.79
iCaRL(our impl.)	100%	36.14	26.26	-4.89	60.80	36.41	-0.60
w/ Efficient-CLS	25%	38.36(+2.22)	26.64(+0.38)	-8.20(-3.31)	61.26(+0.46)	39.58(+3.17)	-3.48(-2.88)
	100%	40.24(+4.10)	28.18(+1.92)	-8.10(-3.21)	67.05(+6.25)	40.36(+3.95)	-3.67(-3.07)
A-GEM	100%	36.94	26.19	-5.54	58.79	35.88	-8.38
w/ Efficient-CLS	25%	37.06(+0.12)	26.36(+0.17)	-7.76(-2.22)	63.06(+4.27)	39.46(+3.58)	-7.49(+0.89)
	100%	39.87(+2.93)	27.97(+1.78)	-7.17(-1.63)	66.94(+8.15)	39.57(+3.69)	-11.68(-3.30)
GDumb	100%	35.27	25.29	-6.59	58.85	36.38	-5.21
w/ Efficient-CLS	25%	37.67(+2.40)	25.59(+0.30)	-9.30(-2.71)	62.70(+3.85)	38.78(+2.40)	-8.86(-3.65)
	100%	38.61(+3.34)	26.04(+0.75)	-9.14(-2.55)	63.55(+4.70)	38.98(+2.60)	-7.50(-2.29)
DER++	100%	37.79	25.24	-2.87	55.82	30.84	-6.08
w/ Efficient-CLS	25%	37.93(+0.14)	25.64(+0.4)	-8.90(-6.03)	59.70(+3.88)	34.15(+3.31)	-11.21(-5.13)
	100%	39.61(+1.82)	26.73(+1.49)	-8.30(-5.43)	62.01(+6.19)	33.09(+2.25)	-11.05(-4.97)

5 RESULTS

5.1 PERFORMANCE IN FULLY SUPERVISED PROTOCOL

As the previous work (Wang et al., 2021a) focuses on online continual object detection (OCOD) in video streams, we first evaluated model performance in fully supervised setting (*i.e*., 100% annotation cost), where all video frames are paired with ground truth labels. We reported the results measured by standard metrics (CAP, FAP, and F, Section 4.3) in Table 1.

Comparisons with previous SOTA. Previously, Wang et al. (2021a) benchmarked Incremental, EWC, iCaRL(Wang *et al*.), and Offline Training on OAK dataset. They found the replay-based method (*i.e*. iCaRL(Wang *et al*.)) outperforms regularization-based method (*i.e*. EWC) by 10% in FAP, while iCaRL(Wang *et al*.) has a huge gap of 30% compared with Offline Training. Similar observations were made in our setting, but the performance of Incremental and EWC were 4% lower than that in Wang et al. (2021a), as we only trained each mini-batch of video frames once (they trained each mini-batch 10 times). As mentioned in Section 4.2, we introduced several variations to the original design of iCaRL(Wang *et al*.). Compared with iCaRL(Wang *et al*.), we observed a huge performance boost from 22.89% to 36.14% in FAP on OAK and from 37.61% to 60.80% in FAP EgoObjects datasets, abridging the gap between the baseline and Offline Training.

Comparisons with other CL baselines. We further adapted other standard CL baselines, including iOD, A-GEM, GDumb, DER++, to the OCOD setting for comparisons. iOD is the state-of-the-art method in offline class-incremental object detection. Though it performs well in prior setting, iOD collapses when adapted to online video streams. One explanation is that iOD requires explicit task boundary to trigger the reshape of model gradients that optimizes knowledge sharing between adjacent tasks. However, in online video streams, the task boundaries by classes are no longer available and the change of tasks is hard to identify, resulting in the failure to prevent forgetting. In contrast to gradient-based methods like iOD and EWC, replay-based methods such as A-GEM, GDumb, DER++ generalize much better to the real-world streaming video.

Boosting state-of-the-art CL methods. Inheriting from the benefit of fast and slow learning with EMA, our Efficient-CLS consistently improves all the state-of-the-art CL methods (*i.e*. iCaRL(our impl.), A-GEM, GDumb and DER++) by a significant margin. Taking iCaRL(our impl.) for example, with Efficient-CLS, we observed an increase of 4.10% in FAP, 1.92% in CAP % and 3.21% in F on OAK dataset. Since semantic contextual information is more important in indoor environments on EgoObjects compared to the outdoor environments in OAK dataset, we noticed that the improvement brought by Efficient-CLS is even greater on EgoObjects dataset with an increase of 6.25% in FAP, 3.95% in CAP % and 3.07% in F. Consistent performance gains are also noticeable for A-GEM, GDumb and DER++, demonstrating the effectiveness and versatility of Efficient-CLS.

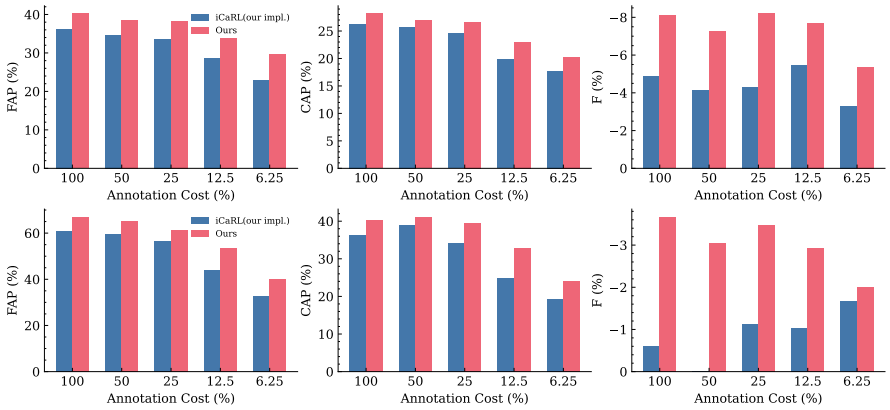


Figure 3: **Evaluation of online continual object detection in video streams with three metrics (FAP, CAP and F, Section 4.3) on OAK dataset (first row) and EgoObjects dataset (second row)**. The higher the bars are, the better. The x-axis denotes the percentage of video frames that are labeled in the video stream. It ranges from 6.25% to 100% (full supervision). The y-axis indcates the performance using different evaluation metrics. Ours (iCaRL(our impl.) w/ Efficient-CLS, red) consistently beats the comparative SOTA (iCaRL(our impl.), blue) in all evaluation metrics.

5.2 Performance in Sparse Annotation Protocol

Comparisons of CL baselines. The sparse annotation protocol is more challenging than the previous fully supervised protocol as shown by the performance differences when number of annotated video frames decreases (compare the performance of each colored bar along the x-axis within each subplot in Figure 3 and Figure S1). We noted that GDumb is more resilient against the reduce of supervision. Specifically, from Table 2 at the lowest annotation cost of 6.25% on EgoObjects dataset, GDumb achieves the highest performance of 38.74%, 22.69% and -4.53% in FAP, CAP and F, which surpasses other baselines by a considerable margin. Same observations can be made on OAK dataset. One possible explanation is that GDumb only trains the data stored in the balanced replay buffer, which makes it less vulnerable to class imbalance problem brought by the reduce of labeled samples in the training set.

Boosting state-of-the-art CL methods. Our proposed Efficient-CLS is a plug-and-play module that can be easily inserted into and improve existing continual learners with the ability to use unlabeled video frames effectively. In both OAK and EgoObjects datasets, Efficient-CLS consistently improves the comparative SOTAs in all three evaluation metrics regardless of various degrees of annotation cost. As shown in Table 2, at a lower annotation cost of 6.25%, Efficient-CLS doubles the performance of DER++ and A-GEM in terms of FAP and CAP, and achieves an even larger improvement in preventing forgetting. Thanks to the useful information from pseudo labels predicted by the slow learner in Efficient-CLS, our method is more robust to various annotation costs, compared with SOTAs (compare the rate of change of blue bars *vs*. red bars over different degrees of annotation cost). Most remarkably, Efficient-CLS with 25% annotation cost has already outperformed comparative SOTAs with 100% annotation cost (see Table 1).

5.3 Ablation Study

We assessed the importance of design choices by evaluating ablated versions of our Efficient-CLS in fully supervised protocol (Table 1) and sparse annotation protocol (Table 3 and Table S3). The complementary learning systems design in Efficient-CLS is the key for rapidly adapting to learn new tasks, meanwhile, retaining previously learnt knowledge. It constitutes of two memory reinstatement mechanisms: one is synaptic weight transfer from fast to slow learner via exponential moving average (EMA); and the other is semantic replay with pseudo-labeling (PL) from slow learner to fast learner. Here we ablated individual mechanism and studied their effect on OAK dataset. We provided additional ablations regarding EMA and PL in Appendix C.

Effect of fast and slow learning with EMA. For brevity, we termed "fast and slow learning with EMA" as "EMA". We removed EMA by setting α to 1 in Equation 3, where the model weights

Table 2: **Effectiveness of Efficient-CLS at annotation cost 6.25% on OAK and EgoObjects**. The metrics are reported in the form of FAP (↑) / CAP (↑) / F (↓).

	OAK		EgoObjects	
	w/o Efficient-CLS	w/ Efficient-CLS	w/o Efficient-CLS	w/ Efficient-CLS
iCaRL	23.04 / 17.75 / -3.31	29.72 / 20.31 / -5.36	32.91 / 19.15 / -1.36	40.16 / 23.95 / -2.01
A-GEM	23.59 / 16.15 / -2.99	30.18 / 20.59 / -5.44	21.84 / 12.28 / 0.82	38.96 / 23.79 / -5.64
GDumb	27.37 / 19.64 / -4.25	29.07 / 19.99 / -6.01	38.74 / 22.69 / -4.53	40.09 / 23.67 / -5.16
DER++	24.21 / 15.93 / -3.79	28.63 / 19.64 / -4.60	16.95 / 8.48 / 2.03	35.78 / 20.74 / -4.69

Table 3: **Ablation of Efficient-CLS on OAK dataset at annotation cost 25%**.

EMA	PL	FAP (↑)	CAP (↑)	F (↓)
✗	✗	33.70	24.57	-4.30
✓	✗	34.79	25.62	-4.35
✗	✓	34.95	25.65	-3.65
✓	✓	**38.36**	**26.64**	**-8.20**

of the fast learner and slow learner are now shared throughout the learning process. Note that in the fully supervised protocol, the pseudo-labeling is turned off and the Efficient-CLS equals to the EMA. From Table 1 at 100% annotation cost, we observed that removing Efficient-CLS leads to a significant performance drop ranging from 2% to 8%, for all the comparative baselines on both OAK and EgoObjects datasets. This shows that the slow learner can effectively consolidate the knowledge from the fast learner, and constructively alleviate catastrophic forgetting by synapses consolidation over time. Given the fact that the slow learner is better at preventing forgetting than the fast learner, we used the output from the slow learner at the inference stage. Similar observations were made on Table 3 and Table S3 in the sparse annotation protocol (compare Row 2 *vs*. Row 1, Row 4 *vs*. Row 3). It is worth noting that, the performance difference between naive model (Row 1) and its variant with EMA (Row 2) is slightly larger in lower supervision (*i.e*. 12.5%, 6.25%) than higher supervision (*i.e*. 50%, 25%) (Table S3). One reason is that compared with higher supervision, the fast learner suffers more forgetting in lower supervision; hence, the effect of removing EMA becomes stronger in lower supervision, again highlighting the importance of EMA.

Effect of semantic replay with pseudo-labeling. For brevity, we termed "semantic replay with pseudo-labeling" as "PL". We ablated our Efficient-CLS by removing the PL of the slow learner across different annotation costs and reported the results in Table 3 and Table S3. The removal of PL (Row 2) leads to a performance drop of around 2% in FAP, 1% in CAP and 0.5-4% in F, compared with our full Efficient-CLS (Row 4). It implies that the slow learner captures useful semantic information from unlabeled video frames and these predicted pseudo labels are helpful in training the fast learner.

To investigate whether pseudo labels predicted by the fast learner itself could help stream learning, we conducted another ablation experiment where we performed PL without EMA (Row 3). Compared with the naive model (Row 1), we observed a performance increase from 28.76% to 31.60% in FAP and 19.80% to 22.44% in CAP. It indicates that, due to the temporal correlation in video stream, pseudo labels predicted by the fast learner can serve as an informative supervision for the training of the fast learner itself. However, replaying the self-predicted pseudo labels on the fast learner fails to prevent forgetting, as indicated by the drop from -5.48% to -4.83% in F. It is possible that the pseudo labels generated by the fast learner only bias towards the classes which have already been learnt very well and fail to reinforce the fast learner to improve on the poorly-learnt classes. Different from the fast learner, the slow learner integrates semantic information over time. The predicted pseudo labels carry more semantic information, which is useful for fast learner to capture more generic object representations during pseudo label replays. Again, this emphasizes that the reciprocal replay from the slow learner to the fast learner is critical for memory reinstatement, which has been missing in the computational modeling literature of CLS.

6 Conclusion

To imitate what humans see and learn in the real world, we introduced a more realistic and challenging problem on label-efficient online continual object detection (LEOCOD) in video streams. Addressing this problem would greatly benefit real-world applications by reducing model retraining time and data labeling costs. Inspired by the complementary learning systems (CLS) in human brains, we proposed a plug-and-play module, namely Efficient-CLS, that can be easily integrated into and improve existing continual learners. We rigorously evaluated Efficient-CLS and competitive baselines on two challenging real-world video stream datasets. We verified the effectiveness and versatility of our method in reducing annotation costs and avoiding catastrophic forgetting. Although our Efficient-CLS only capitalizes on 25% annotations, it beats all comparative models requiring fully supervised training on all video streams.

REFERENCES

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. *Advances in neural information processing systems*, 32, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32, 2019b.

Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint arXiv:2201.12604*, 2022.

Matteo Boschini, Pietro Buzzega, Lorenzo Bonicelli, Angelo Porrello, and Simone Calderara. Continual semi-supervised learning through contrastive interpolation consistency. *Pattern Recognition Letters*, 162:9–14, 2022.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. *arXiv preprint arXiv:2203.03798*, 2022.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Keval Doshi and Yasin Yilmaz. Rethinking video anomaly detection-a continual learning approach. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3961–3970, 2022.

Serena M Dudek and Mark F Bear. Bidirectional long-term modification of synaptic effectiveness in the adult and immature hippocampus. *Journal of Neuroscience*, 13(7):2910–2918, 1993.

Mark Everingham, SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

Enrico Fini, Stéphane Lathuiliere, Enver Sangineto, Moin Nabi, and Elisa Ricci. Online continual learning under extreme memory constraints. In *European Conference on Computer Vision*, pp. 720–735. Springer, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Daoyun Ji and Matthew A Wilson. Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nature neuroscience*, 10(1):100–107, 2007.

Nitin Kamra, Umang Gupta, and Yan Liu. Deep generative dual memory network for continual learning. *arXiv preprint arXiv:1710.10368*, 2017.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Joseph Kj, Jathushan Rajasegaran, Salman Khan, Fahad Shahbaz Khan, and Vineeth N Balasubramanian. Incremental object detection via meta-learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20 (7):512–534, 2016.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3589–3599, 2021.

Quang Pham, Chenghao Liu, Doyen Sahoo, and HOI Steven. Contextual transformation networks for online continual learning. In *International Conference on Learning Representations*, 2020.

Quang Pham, Chenghao Liu, and Steven Hoi. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems*, 34, 2021.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*, pp. 524–540. Springer, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

Farzaneh Rezaeianaran, Rakshith Shetty, Rahaf Aljundi, Daniel Olmeda Reino, Shanshan Zhang, and Bernt Schiele. Seeking similarities over differences: Similarity-based domain alignment for adaptive object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9204–9213, 2021.

Mohammad Rostami, Soheil Kolouri, and Praveen K Pilly. Complementary learning for overcoming catastrophic forgetting using experience replay. *arXiv preprint arXiv:1903.04566*, 2019.

James Smith, Jonathan Balloch, Yen-Chang Hsu, and Zsolt Kira. Memory-efficient semi-supervised continual learning: The world is its own replay buffer. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Jianren Wang, Xin Wang, Yue Shang-Guan, and Abhinav Gupta. Wanderlust: Online continual object detection in the real world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10829–10838, 2021a.

Liyuan Wang, Kuo Yang, Chongxuan Li, Lanqing Hong, Zhenguo Li, and Jun Zhu. Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5383–5392, 2021b.

Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

Aming Wu, Rui Liu, Yahong Han, Linchao Zhu, and Yi Yang. Vector-decomposed disentanglement for domain-invariant object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9342–9351, 2021.

Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *European conference on computer vision*, pp. 566–583. Springer, 2020.

SUPPLEMENTARY MATERIAL

A EXPERIMENTAL SETUPS

A.1 TRAINING

For a fair comparison, we follow the prior work (Wang et al., 2021a) to use Faster-RCNN (Ren et al., 2015) with ResNet-50 backbone (He et al., 2016) as our object detection network, which is initialized by the weights pre-trained on PASCAL VOC (Everingham et al., 2015). We use Adam optimizer with a learning rate 0.0001, and the batch size is set to 16 frames. Same as Wang et al. (2021a), we maintain a replay buffer with 5 samples per class (see Appendix A.2). At each time step t, we first randomly retrieve 16 video frames from the replay buffer for joint training. We use confidence thresh $\tau = 0.7$ to generate pseudo-labels for unlabeled frames. Data augmentation, including random image crops, rotations, and horizontal flip, is applied on these pseudo-labeled frames. We introduce $\lambda_{pseudo} = 1.0$ as a hyper-parameter to balance the contribution of two losses $\mathcal{L}_{sup}$ and $\mathcal{L}_{pseudo}$. After updating the weights of the fast learner via backpropagation of the incurred losses, we update the slow learner by taking the EMA of the fast learner's weights with an EMA rate $\alpha = 0.99$. Finally, the replay buffer is updated with the labeled frame at current time step t. Each model is trained by a single pass over the entire video stream. The training is carried out on 2 NVIDIA RTX 3090 GPUs.

A.2 Implementation of Replay Buffer

Briefly, we follow Wang et al. (2021a) to maintain a balanced replay buffer with 5 sample images per class. Intuitively, the replay buffer acts as a list of arrays, where the length of the list is equal to total number of learnt classes and each array stores 5 sample images containing that class. It is possible that any given image contains multiple object classes. In this case, we will randomly select the "representative" class and store this image in its relevant class array of the replay buffer. Note that all the bounding box annotations and their corresponding class labels on the given image would also get stored. When the replay buffer is full and there is a new image in the video stream, Efficient CLS will randomly decide whether the image will be saved to the buffer with a fixed probability 5/6 and if so, which stored element in the class array will be replaced with the given image of probability 1/5. To summarize, there are at least 5 images containing object instances of any given learnt classes. The replay buffer is NOT part of the network architecture and it is NOT fully differentiable. The total replay buffer size is around 500 frames for OAK, and 1400 for EgoObjects. To perform episodic replay, we randomly retrieve 16 video frames from the replay buffer for joint training with current frames of a mini-batch. We also justify these design decisions with additional ablation studies on balanced buffer, buffer size, and replay size in Appendix C.7 and provide results without replay in Appendix B.1.

A.3 Evaluation Metrics

Following Wang et al. (2021a), we evaluate all the methods with three standard metrics: continual average precision (CAP), final average precision (FAP) and forgetfulness (F). We adopt AP50, *i.e.*, the average precision (AP) at IoU $= 0.5$, as the measurement of AP.

CAP shows the average performance of a continual learning algorithm over the time span of the entire video stream. As shown in Figure 1, the model is evaluated on the test set every 100 time steps. At i^{th} evaluation step, the reported CAP_{t_i} is defined as

$$\mathrm{CAP}_{t_i} = \frac{1}{C}\sum_{c=0}^{C}\mathrm{CAP}^c_{t_i}, \tag{S1}$$

where $\mathrm{CAP}^c_{t_i}$ is the average precision (AP) of the class c on the test set. CAP is then defined as the average values over all the evaluation steps:

$$\mathrm{CAP} = \frac{1}{N}\sum_{i=0}^{N}\mathrm{CAP}_{t_i} = \frac{1}{NC}\sum_{i=0}^{N}\sum_{c=0}^{C}\mathrm{CAP}^c_{t_i}, \tag{S2}$$

where N is the total evaluation steps.

FAP is the final performance of a model after seeing the entire video. That is, $\mathrm{FAP} = \mathrm{CAP}_{t_N}$, where t_N denotes the last evaluation step.

F estimates the forgetfulness of the model due to the sequential training. It takes into account the time interval between the presence of an object category and its subsequent presence. For a class c, we sort the $\mathrm{CAP}^c_{t_i}$ according to the time interval k between evaluation time t_i and the last time $t_i - k$ the model is trained on c. After $\mathrm{CAP}^c_{t_i}$ is sorted, all $\mathrm{CAP}^c_{t_i}(i = 0, \cdots, T)$ are divided into K bins $B_{kmin}, \cdots, B_{kmax}$ according to the time interval k. The average CAP (aCAP_k) of each bin B_k is defined as the model's performance for detecting class c after the model has not been trained on c for k time steps. The forgetfulness (F) of the class c is defined as the weighted sum of the performance decrease at each time:

$$\mathrm{F}^c = \sum_{k=kmin}^{kmax}\frac{k - kmin}{\sum_{k=kmin}^{kmax} k - kmin} \times (\mathrm{aCAP}_{kmin} - \mathrm{aCAP}_k). \tag{S3}$$

The overall forgetfulness is then defined as:

$$\mathrm{F} = \frac{1}{C}\sum_{c=0}^{C}\mathrm{F}^c \tag{S4}$$

A.4 DATASETS

OAK. We follow Wang et al. (2021a) in the ordering of training and testing data splits, *i.e.*, one frame every 16 consecutive video frames is held out to construct a test set and the remaining frames are used for training. However, as the original test set curated by Wang et al. (2021a) is not publicly available, we re-split the training and testing data using the video streams from the original training set. The model trained and evaluated on our dataset shows comparable results with the original one.

EgoObjects. The original data can be downloaded from this website[4]. This dataset consists of 6076 videos taken in 1110 realistic indoor environments (around 6 videos per environment). The videos in each environment contain the same objects but feature a great variety of lighting conditions, scale, camera motion, and background complexity. We first downsample the original videos by 2 frames to make the length of the entire video stream comparable with OAK dataset. We shuffle the ordering of 6076 videos (not the video frames from the same video) from different environments to make it more realistic as the previously seen environments are allowed to be revisited in real-world. We concatenate the videos as one long video stream, and use the same ordering as OAK to construct the train and test data splits.

B ADDITIONAL RESULTS AND DISCUSSIONS

B.1 PERFORMANCE OF EFFICIENT-CLS WITHOUT REPLAY

Table S1: **Performance of Efficient-CLS and other state-of-the-art methods without episodic replay on OAK dataset.**

	Annotation Cost	FAP (↑)	CAP (↑)	F (↓)
iCaRL(our impl.) w/o replay	100%	7.70	8.03	0.72
	50%	6.87	7.19	0.97
	25%	6.02	5.41	1.13
	12.5%	4.56	5.09	-0.02
	6.25%	3.43	3.25	-0.26
Efficient-CLS w/o replay (EMA only)	100%	9.79	14.09	-2.13
	50%	8.63	12.15	-1.54
	25%	7.54	9.89	-1.82
	12.5%	5.95	8.00	-1.45
	6.25%	4.84	6.29	-1.32
Efficient-CLS w/o replay (EMA+Pseudo-labeling)	100%	9.79	14.09	-2.13
	50%	8.45	11.93	-1.98
	25%	6.36	8.53	-1.39
	12.5%	4.52	5.80	-0.98
	6.25%	2.63	2.97	-0.17

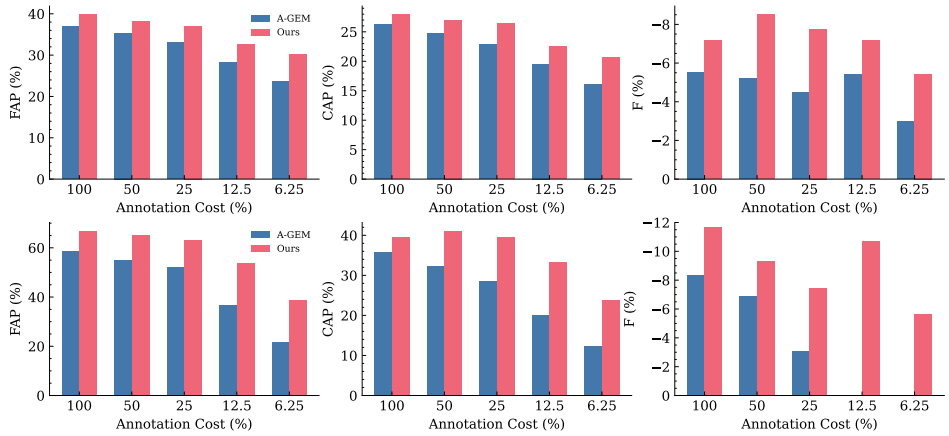
We reported the results of our Efficient-CLS w/o replay in both fully supervised and sparse annotation protocols:

Fully Supervised Protocol. As shown in Table S1, the Efficient-CLS w/o replay still achieves the state-of-the-art performance in the fully supervised protocol, demonstrating the effectiveness of slow-fast learning with EMA. Specifically, our method outperforms the iCaRL baseline by 6.37% in FAP, 1.41% in CAP, and 2.16% in forgetting when the external memory is removed.

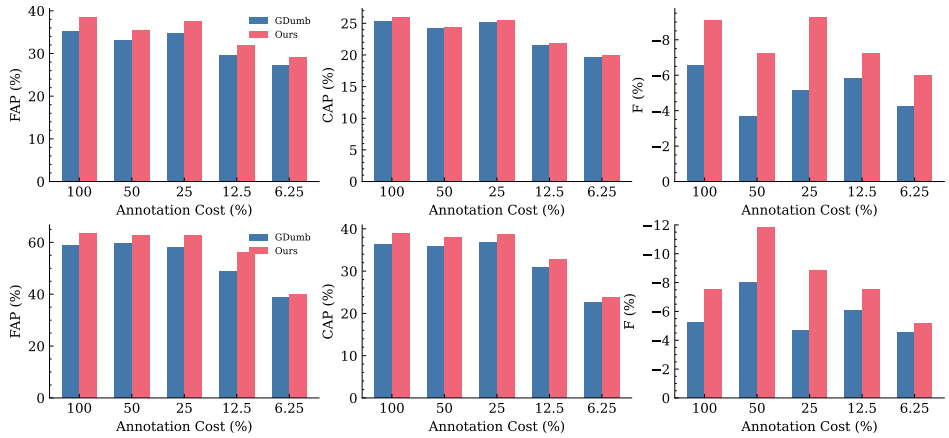
Sparse Annotation Protocol. When there is no replay, the slow-fast learning with EMA in our method can still lead to significant improvement; while the episodic replay is critical for generating meaningful pseudo labels. Detailed results as follows:

- *The effect of EMA*: In Table S1, we observed a significant improvement of our slow-fast learning mechanism over the baseline model iCaRL(our impl.) w/o replay over all annotation costs. For example, at annotation cost of 50%, Efficient-CLS w/o replay (EMA only) surpasses iCaRL without replay by around 5% in FAP, 2% in CAP, and 2% in Forgetting.

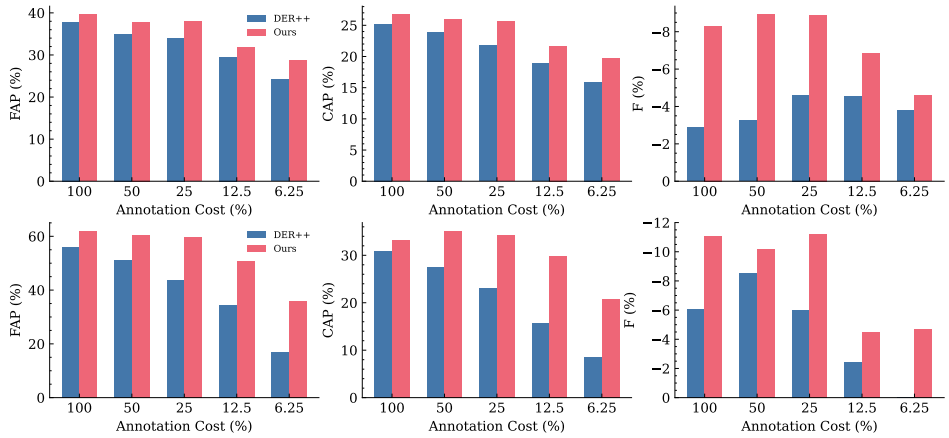
[4]https://sites.google.com/view/clvision2022/challenge



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.

- *The effect of pseudo-labeling*: When the pseudo labels generated by the slow learner are applied for joint training, we found the performance drops as the amount of annotations decreases, even resulting in inferior results to the baseline model at 6.25% annotation cost. This is opposite with the experiments using episodic replay. We conjecture that the poor performance of the slow learner could lead to inaccurate and biased pseudo labels, which exacerbates the model learning.

Overall, we showed that both episodic replay and EMA are important for LEOCOD. Moreover, in the sparse annotation scenario, the episodic replay is critical in encouraging slow-learners to generate meaningful pseudo labels, which can further improve the performance of Efficient-CLS.

B.2 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 S2. We found that our Efficient-CLS shows reliable and robust performance against different selections of unlabeled frames in the video stream.

Table S2: **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.

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)

B.3 Analysis of AP Changes over Time

As shown in Figure S2, our method, *i.e.* iCaRL(our impl.)+Efficient-CLS (light blue), consistently outperforms state-of-the-art approaches with minimal forgetting even when categories appear infrequently (*e.g.*, sculpture) and exhibits the closest gap against the upper bound, *i.e.* Offline (purple).

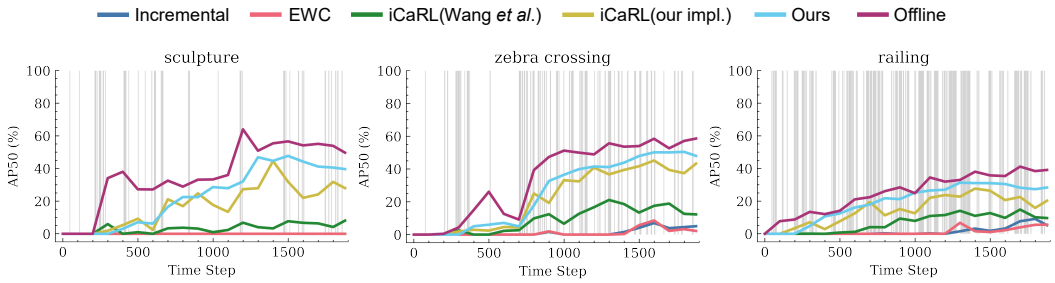


Figure S2: **The changes of $\mathrm{CAP}^c_{t_i}$ with sampled categories on OAK dataset in fully supervised protocol.** The x-axis denotes the time step across the entire video stream. The y-axis denotes the AP50 of the category at specific time step (*i.e.*, $\mathrm{CAP}^c_{t_i}$). The grey line indicates the existence of the category.

B.4 Analysis of Inference Model

Figure S3 shows that the slow learner (orange) of our Efficient-CLS is better at preventing forgetting than the fast learner (blue), hence we used the slow learner at inference stage (Section 5.3).

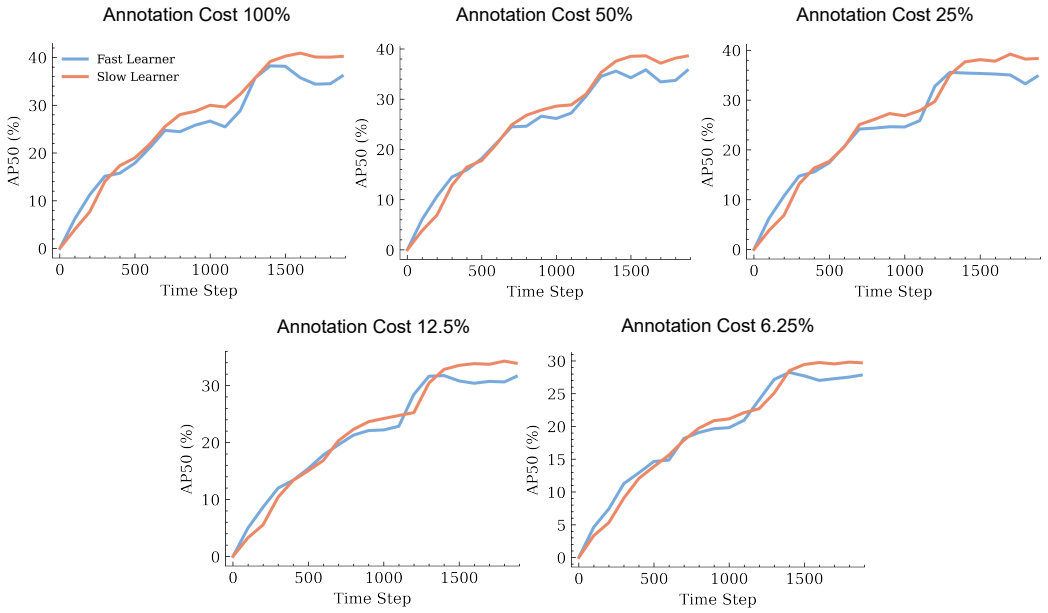


Figure S3: **The changes of CAP_{t_i} on OAK dataset at different annotation costs.** The x-axis denotes the time steps across the entire video stream. The y-axis denotes the AP50 at specific time step (*i.e.*, CAP_{t_i}).

B.5 Visualization of Pseudo-labeling

As shown in Figure S4, 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.6 Feasibility Analysis of LEOCOD

Note that in the area of online continual learning, both mainstream works in image domain (Aljundi et al., 2019b; Mai et al., 2021) and video domain (Doshi & Yilmaz, 2022; Wang et al., 2021a) require humans in the loop to label each data sample before processing the next sample. Thus, we follow this standard setting. Yet, it might be challenging to have humans in the loop to provide real-time annotations. In particular, Wang et al. (2021a), which is the first online continual learning work for video object detection and most relevant to our paper, requires labeling every frame and hence not realistic.

Our LEOCOD setting is specifically designed to make the current setting of online video continual learning more realistic, because our human annotators label only frames that are sparsely sampled. Note that there are some annotation companies that do provide such stand-by online annotation services and can even have multiple annotators to simultaneously label the same frame but each annotator in charge of different classes. On the OAK dataset:

- The gap between every 2 consecutive frames is 2s (Wang et al., 2021a), our model will be updated every mini-batch of 16 frames (Figure 1), so the time we have for annotation before processing the next batch is 2s x 16 = **32s**.
- We repeat OAK's annotation process and find that each frame takes on average 45s to label. Given 3 annotators simultaneously labeling the same frame, our wall-clock annotation time needed for each frame is 45s / 3 = **15s**.
- If our sparse sampling rate is 12.5% i.e. 2 frames per batch, the wall-clock annotation time for ours is 15s x 2 = **30s** which is within the tolerance of 32s, while Wang et al. (2021a) needs 15s x 16 = **240s**.

Figure S4: **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.

Our LEOCOD task is important for many applications that target high accuracy, such as self-driving car, caring-robot for elderly. In these applications, our setting of requiring human labels could significantly help

- alleviate domain shift (Rezaeianaran et al., 2021; Wu et al., 2021) where old object detectors fail;
- alleviate catastrophic forgetting of old classes;
- improve accuracy of new class compared to learning w/o human labels.

B.7 Detailed Discussions on Neuroscience Inspirations

Cognitive science works Wang et al. (2020); Lake et al. (2017) show that humans are efficient at continuously learning from very few annotated data samples. We get inspirations from the theory of Complementary Learning Systems (CLS) in human brains, and propose a general framework for Label-Efficient Online Continual Object Detection, dubbed as Efficient-CLS model. In particular, CLS theory postulates that the memories are first encoded via fast synaptic changes in the hippocampus and then these changes support slow reinstatement of memories in the neocortex via accumulated experiences over time Kumaran et al. (2016). To mimic the fast and slow synaptic changes in hippocampus and neorcotex, in Efficient-CLS, we introduce two feed-forward neural networks as slow and fast learners. In the fast learner, memory is encoded in its synaptic weights and these weights adapt rapidly to the current task. The synapses of the slow learner change a little on each reinstatement, and are maintained by taking the exponential moving average of the fast

learner's synapses over time. Though a few continual learning models in previous works Arani et al. (2022); Pham et al. (2020) also use a similar source of inspiration, they miss the effect of reciprocal connections from neocortex to hippocampus, which we intend to address. The neuroscience study Ji & Wilson (2007) has identified the importance of bidirectional interaction between the two complementary systems, whereby the reactivation of neural patterns in the neocortex triggers replays in the hippocampus, which in turn drive the memory consolidation in the neocortex. Inspired by this underlying mechanism, we reactivate the weights of the slow learners to predict meaningful pseudo labels from the unlabeled video frames and use these pseudo labels to guide the training of the fast learner, closing the loop between the two complementary learning systems. Specifically, pseudo labels, predicted by the slow learner, carry integrated semantic information over time, which encourages the fast learner to capture more holistic scene representations, alleviating the catastrophic forgetting problem on sparsely annotated videos.

B.8 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.

C Additional Ablations

In addition to the ablation studies provided in the main paper, we further studied the effectiveness of each component in our proposed Efficient-CLS in the following sections.

C.1 Effect of EMA and Pseudo-labeling

In the main text, we studied the effect of EMA and pseudo-labeling at 25% annotation costs (Section 5.3). Here we performed the full experiments at all proportions of annotation cost in Table S3. We found that, by integrating the EMA and pseudo-labeling, our Efficient-CLS (4th row) improves the state-of-the-art model (1st row) and other ablated models (2nd row and 3rd row) by a significant margin. Please refer to Section 5.3 for more analysis.

Table S3: **Effectiveness of Exponential Moving Average (EMA) and Pseudo-labeling (PL) for iCaRL(our impl.) on OAK dataset at annotation cost 50%, 25%, 12.5% and 6.25%**. The best results are **bold-faced**.

		50%			25%			12.5%			6.25%		
EMA	PL	FAP (↑)	CAP (↑)	F (↓)	FAP (↑)	CAP (↑)	F (↓)	FAP (↑)	CAP (↑)	F (↓)	FAP (↑)	CAP (↑)	F (↓)
✗	✗	34.68	25.78	-4.15	33.70	24.57	-4.30	28.76	19.80	-5.48	23.04	17.75	-3.31
✓	✗	35.74	25.77	-4.82	34.79	25.62	-4.35	31.72	21.16	-7.24	27.84	20.03	-3.96
✗	✓	35.61	25.56	-3.76	34.95	25.65	-3.65	31.60	22.44	-4.83	26.39	19.50	-1.99
✓	✓	**38.61**	**26.90**	**-7.29**	**38.36**	**26.64**	**-8.20**	**33.92**	**23.04**	**-7.71**	**29.72**	**20.31**	**-5.36**

C.2 Effect of EMA Rates

To further explore the role of EMA, we varied α from 0.5 and 0.999 and presented their performance in Table S4. We observed that the choice of α is relatively insensitive to the performance. For example, Efficient-CLS with $\alpha = 0.9$ leads to an performance increase of less than 1% in FAP and CAP and 1.12% in F, compared with the case of $\alpha = 0.5$. However, we did observe a huge performance drop when α is very close to 1.0, where there is almost no synaptic weight transfer between slow and fast learner.

C.3 Effect of Pseudo-labeling Threshold

As mentioned in Section 3.2.2, we apply confidence thresholding to remove predicted bounding boxes that have low confidence scores. To show the effectiveness of thresholding, we varied the

Table S4: **Ablation study of varying EMA rates α on OAK dataset in fully supervised protocol.**

α	0.5	0.9	0.95	0.99	0.995	0.999
FAP (↑)	36.93	37.57	38.25	40.24	40.59	33.02
CAP (↑)	26.70	28.35	28.61	28.18	27.11	15.15
F (↓)	-6.66	-5.54	-6.20	-8.10	-9.70	-5.72

confidence threshold τ from 0.1 to 0.9 (see Table S5). 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.

Table S5: **Ablation study of varying confidence threshold τ at annotation cost 12.5%.**

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

C.4 Effect of RPN Loss in Pseudo Training

In Section 3.2.2, we mentioned that pseudo losses are only applied at the ROI module but not at the RPN module. As shown in Table S6, 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.

Table S6: **Performance of our Efficient-CLS with and without RPN loss in pseudo training on OAK dataset at annotation cost 12.5%**. The best results are **bold-faced**.

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

C.5 Effect of Pseudo Loss Weights

As mentioned in Section 3.2.2, 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 S7, 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.

Table S7: **Ablation study of varying pseudo loss weights λ_{pseudo} at annotation cost 12.5%.**

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

C.6 Effect of Data Augmentation in Pseudo Training

As mentioned in Section 3.2.2, we use data augmentation techniques when training the pseudo-labeled frames. Here we ablated our Efficient-CLS by removing data augmentation in pseudo train-

ing. From Table S8 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.

Table S8: **Effectiveness of Data Augmentation in Pseudo Training on OAK dataset at annotation cost 12.5%**. The best results are **bold-faced**.

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

C.7 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 S9, we found that a class balanced replay buffer performs much better than Random Store and Replay, implying the importance of class balanced replay buffer.

Table S9: **Effectiveness of class balanced replay buffer on OAK dataset in fully supervised protocol.** The best results are **bold-faced**.

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

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 S10). 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 S10, 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. Wang et al. (2021a) 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.)).

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 S11). 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.

Table S10: **Ablation study of varying numbers of replay samples per training iteration on OAK dataset in fully supervised protocol.** The buffer size is set to 5 images per class. Our choices are **bold-faced**.

	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 S11: **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**.

	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