Studying Forgetting in Faster R-CNN for Online Object Detection: Analysis Scenarios, Localisation in the Architecture and Mitigation

Baptiste Wagner, Denis Pellerin, Sylvain Huet

Studying Forgetting in Faster R-CNN for Online Object Detection: Analysis Scenarios, Localisation in the Architecture and Mitigation

BAPTISTE WAGNER, DENIS PELLERIN and SYLVAIN HUET
Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab, 38000 Grenoble, France

Corresponding author: Baptiste Wagner (e-mail: baptiste.wagner@gipsa-lab.grenoble-inp.fr).

ABSTRACT Online Object Detection (OOD) requires learning novel object categories from a stream of images, similarly to an agent exploring new environments. In this context, the widely used Faster R-CNN architecture faces catastrophic forgetting: acquiring new knowledge leads to the loss of previously learned information. In this paper, we investigate the learning and forgetting mechanisms of the Faster R-CNN in OOD through three main contributions. Firstly, We show that the Faster R-CNN's forgetting curves reflect human memory cognitive processes as developed by Hermann Ebbinghaus: knowledge is lost exponentially over time and recalls enhance knowledge retention. Secondly, we introduce a new methodology for analysing the Faster R-CNN architecture and quantifying forgetting across the Faster R-CNN components. We show that forgetting is mainly localized in the Softmax classification layer. Lastly, we propose a new training strategy for OOD called Configurable Recall (CR). CR performs recalls on old data using images stored in a memory buffer with variable frequency and recall length to ensure efficient learning. CR also masks the logits of old objects in the Softmax classification layer to mitigate forgetting. We evaluate our strategy against state-of-the-art methods across three OOD benchmarks. We analyze the effectiveness of different recall types in mitigating forgetting and show that CR outperforms existing methods.

INDEX TERMS Catastrophic Forgetting, Ebbinghaus Forgetting Curve, Faster R-CNN, Natural Replay, Online Continual Learning, Online Object Detection,

I. INTRODUCTION

The foundation of intelligent systems lies in their ability to adapt and learn continuously from ever-changing environments. Just as humans have evolved to acquire, update, and utilize knowledge dynamically, we aspire for artificial intelligence systems to exhibit similar adaptability [1], [2].

Online Continual Learning (OCL) focuses on training models to learn continuously from a stream of data, adapting to new information while retaining previously learned knowledge [3]–[5]. The implementation of these systems has the potential to improve the advancement of intelligent technologies operating in dynamic environments, such as autonomous vehicles [6], [7], robotics [8], [9], VR/AR headsets [10] and real-time surveillance [11], [12]. One of the key challenges in OCL is mitigating the problem of *catastrophic forgetting* [13], [14], where a model loses previously acquired knowledge when trained on new data.

The existing literature focuses on studying catastrophic forgetting in an OCL context for classification tasks [3], [15]–[23]. However, few works investigate the phenomenon of forgetting in object detection tasks during learning from an online data stream, which is called Online Object Detection (OOD). [24]–[27]. OOD scenarios emulate realistic data acquisition conditions by mimicking how an agent wanders through new environments [24]. An agent might revisit the same places and repeatedly encounter previously seen objects throughout its lifespan. This phenomenon of revisiting old objects in the data stream is called Natural Replay (NR) and acts as recalls on old data [27].

One of the most widely used architectures for solving the object detection task is Faster R-CNN [28]. However, Faster R-CNN is prone to catastrophic forgetting when learning new objects in an incremental manner [29], [30]. The adaptation of this architecture to the OOD context remains poorly explored [25], [26]. In addition, there has been no research to identify which components of the architecture are subject to forgetting. Understanding these elements are key to propose effective architectural modifications to mitigate forgetting and improve the model's robustness in long-term, real-world applications [31].

Following a review of related work in Section II, our study makes the following three key contributions to the field of OOD:

1) **Impact of Recalls on Learning and forgetting:** NR acts as recalls of old data within the stream. The extent to which it helps the model in memorize old objects and

moderate forgetting remains unclear. In Section III, we explore the sensitivity of the Faster R-CNN architecture to these recalls. We analyse forgetting phases where no NR is present and we trigger recalls with varying lengths and periods to observe the model's responses. Our empirical investigations draw parallels between the learning and forgetting processes in the Faster R-CNN artificial intelligence model and human cognitive processes, as understood through Ebbinghaus's seminal work on human memory [32].

2) **Forgetting Localisation in Faster R-CNN:** If data on older objects is not available, the model may forget how to recognise previously learned objects. [33]. The mechanisms of the forgetting process within the architecture remain poorly understood. In Section IV, we analyze the extent to which each component of the architecture loses knowledge in an OOD learning setup. To achieve this, we introduce a new protocol called Module Probing, to investigate and locate where forgetting occurs within the Faster R-CNN architecture.
3) **New strategies to remediate forgetting for OOD:** Based on our results and insights into the model's response to recalls and the localisation of forgetting within the Faster R-CNN architecture, we propose in Section V a novel strategy called Configurable Recall (CR). CR consists of two key components that work together to address forgetting. First, it uses an external memory buffer to store old images and perform periodic recalls on these images to reduce forgetting. Second, we enhance the Faster R-CNN with a loss function that compensates for the absence of non-revisited objects. Finally, we compare our method with different baselines and evaluate its effectiveness.

II. RELATED WORK

A. ONLINE CONTINUAL LEARNING

Catastrophic forgetting [13], [14] has led to the emergence of the field of Continual Learning [34]–[36]. In the context of classification tasks, several strategies have been proposed to mitigate this issue, including replay mechanisms [15], [17], [18], [37]–[39], regularization techniques [40]–[44], and dynamic architectural adjustments [45]–[47]. A specific paradigm called Online Continual Learning (OCL), it consists in learning new classes from an unshuffled data stream in a single pass [3], [48]. The primary strategies employed to mitigate catastrophic forgetting in OCL use an external memory buffer [3], [15]–[17], [19], [20], [49]. The memory buffer stores a subset of previously learned data samples. When learning new information, the model periodically revisits the stored samples to reinforce old knowledge. This helps in maintaining the model's performance on past tasks while learning new ones. Nevertheless, the majority of research in OCL is concentrated on the classification task [3], [48]. This study addresses the more complex OOD problem, which involves both detection and classification tasks [24].

B. FASTER R-CNN ARCHITECTURE

Various architectures have been proposed for object detection tasks, with Faster R-CNN being one of the most prominent due to its optimal balance between accuracy and speed [28]. Other notable architectures include YOLO (You Only Look Once) [50] and SSD (Single Shot MultiBox Detector) [51]. Our work focuses on the Faster R-CNN architecture because of its extensive use and robust performance in a wide range of applications.

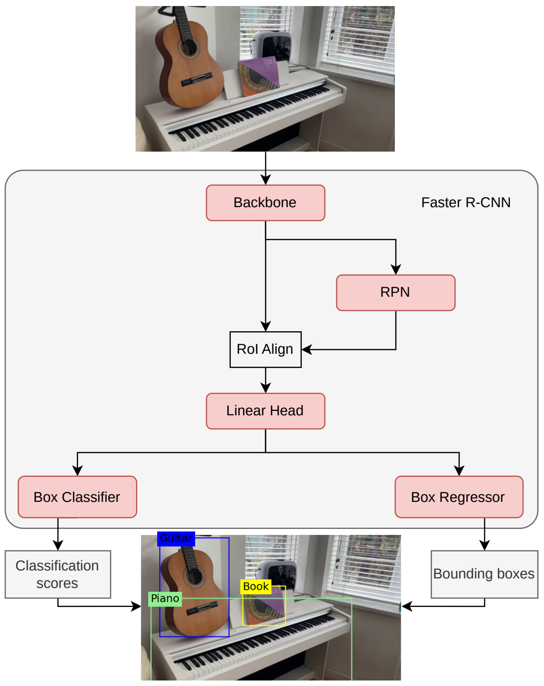


FIGURE 1. **The architecture of Faster R-CNN. An input image (from EgoObjects [52]) is fed in a Backbone to produce a global image feature map. Then, the RPN (Region Proposal Network) proposes region of interest where objects lies in. For each region, the RoI Align operation generates a region specific feature map. Finally, after two fully connected layers in the Linear Head, a softmax layer outputs the classification score in theBox Classifier, and the regions are refined in the final bounding boxes via a regression layer in the Box Regressor.**

Faster R-CNN is part of the two-stage detectors family: the first stage localizes objects by proposing Regions of Interest (RoI) and the second stage classifies the detected objects. Its overall architecture is schematized in Figure 1. The following modules constitute the Faster R-CNN.

Backbone: extracts features from the entire input image by combining a Convolutional Neural Network (CNN) with a Feature Pyramid Network (FPN). This step provides a global image feature for subsequent modules.

Region Proposal Network (RPN): proposes Regions of Interest (RoI) in the form of bounding boxes around potential objects in the image. These proposals serve as a basis for object localization.

RoI Align: generates a feature map for each Region of Interest given by the RPN. Specifically, it processes the image features along with a RoI as input, generating a RoI feature of fixed size, regardless the size of the bounding box predicted by the RPN.

Linear Head: reduces the dimension and flattens the feature map outputed by the RoI Align. This module is composed of two fully connected linear layers.

Box Classifier: assigns a classification score for each detected object given the features extracted from the Linear Head. This module is a Softmax classification layer. The logit with the highest score determines the predicted class.

Box Regressor: refines the region of interest to obtain a more accurate bounding box around the detected object via linear regression.

Indeed, this architecture is famously used in different problems, including multi-task, open-world, meta-learning, and few-shot learning. In this study, the training paradigm is OOD, where the Faster R-CNN is trained on a data stream in a one-pass manner with varying distribution [24]. The main issue is how to counteract catastrophic forgetting when learning new data.

C. ONLINE OBJECT DETECTION

Most recent studies on Continual Object Detection adapted pre-existing static datasets like Pascal VOC [53] and COCO [54] with multiple passes on the data. In this setup, many studies have demonstrated that the Faster R-CNN is prone to catastrophic forgetting when trained on a sequence of data [30], [55]–[58]. Our work focuses on OOD, where training data is presented as a stream, and the model has access to only a single pass for training. Three benchmarks are commonly used to evaluate OOD models [24], [26], [27]:

OAK [24]: This benchmark was introduced in a pioneering work on OOD [24]. It features an egocentric video stream covering nine months in the life of a graduate student, capturing diverse outdoor scenes across a university campus. This dataset is a subset of the KrishnaCam dataset, comprising approximately 7.6 million frames divided into 460 video clips, with a total duration of 70.2 hours. The dataset includes 105 object categories and provides exhaustive bounding box annotations for 80 video snippets, totaling around 17.5 hours.

EgoObjects CL Instance [52]: This benchmark contains 100K images with 250K box annotations for 1.1K object instances. Annotations are exclusively provided for the main object instance in each image, with the instance ID serving as the class label for prediction. The dataset offers a diverse range of backgrounds, surrounding objects, distances, lighting conditions, and camera motions, providing a rich and varied environment for object detection tasks.

EgoObjects CL Category [52]: The same image data stream as EgoObjects CL Instance is used. In the Category benchmark, all objects in each image are annotated. The detector must predict the category class from 277 different object categories.

In terms of data splits, OAK adopts a specific strategy to facilitate the evaluation of OOD models. One frame is held out every 16 labeled frames to construct an test set, while the remaining frames are used for training. This method ensures that the training and test sets cover similar time ranges over the nine months, allowing for realistic assessment of catastrophic forgetting and the model's ability to learn continuously from the data stream.

The two benchmarks on EgoObjects are originally introduced for continual object detection [52]. To respect the *online constraint* specific to the OOD field of study, some works [24], [26], [27] have followed the original data apparition order from the benchmark but constrained training to a *single epoch*. In addition, they built a test set similar to OAK by performing a split that holds out one frame every 16 labeled frames. In this study, we employ these three benchmarks to ensure comprehensive assessment.

In OOD, the system must recognize and locate objects in images presented as a continuous stream of data. Formally, an OOD *scenario* $\mathcal{D}$ is defined as an ordered sequence of T *experiences* as $\mathcal{D} = \{D_1, D_2, \ldots D_T\}$. Each experience D_i is a dataset composed of a set of annotated images. In each image, there is at least one annotated object that the model must learn how to detect and classify. Training is performed online: the model processes data batch by batch and does not have no access to previous batches once they have been processed.

D. NATURAL REPLAY

In OOD scenarios, the same object can appear in different images from separate experiences. Revisiting objects in a Continual Learning setting has been explored in the context of classification [59]–[61]. In a previous study [27], we defined Natural Replay (NR) as the concept of revisiting objects in an OOD data stream. As NR provides a more realistic data stream for real-world learning, we showed that it can obscure the evaluation of forgetting. Indeed, good performance on an object does not solely indicate the model's intrinsic ability to retain information; it can result from frequent exposure to the object in the data stream. Instead of measuring forgetting, there is a risk of measuring the amount of repeated recall exposure from the data stream. Consequently, as object categories exhibit varying levels of NR, the evaluation of model performance becomes biased towards more frequently replayed classes.

To identify which objects benefit from repeated exposure and to quantify NR in an OOD scenario, we introduced in our previous study [27] the Natural Replay Rate (NRR) and Natural Replay Score (NRS) metrics. The NRR metric measures the degree to which an object is revisited in the stream, specifically defined as the index of dispersion for object occurrences across all experiences [62]. Additionally, the NRS metric evaluates the overall NR in the scenario as the average NRR over all objects.

$$NRR(y) = \frac{T((\sum_{i=1}^{T} occ_i(y))^2 - \sum_{i=1}^{T} occ_i(y)^2)}{(T-1)(\sum_{i=1}^{T} occ_i(y))^2} \quad (1)$$

$$NRS = \frac{1}{C}\sum_{y=1}^{C} NRR(y) \quad (2)$$

where T is the number of experiences in the scenario, $occ_i(y)$ is the number of occurrences of object class y in experience D_i, and C is the number of object categories in the scenario.

Benchmarks presented in section II-C EgoObjects CL Instance, EgoObjects CL Category, and OAK exhibits respectively a NRS of 0, 0.51, and 0.92.

In Sections III and IV, we conduct a comprehensive analysis with different experiments of the Faster R-CNN model's forgetting process for previously learned objects. Thus, we chose to conduct our experiments using the EgoObjects CL Instance benchmark. Among the three benchmarks used for OOD in recent works, this is the only one that presents no NR, with an NRS of 0. The scenario in this benchmark comprises $T = 5$ experiences, each disjoint in terms of objects. Consequently, objects present in D_1 do not reappear in later experiences $D_2, \ldots, D_5$. This setup provides an ideal environment for measuring model forgetting.

III. RECALLS EFFECTS ON LEARNING AND FORGETTING

In real-world scenarios, an agent moving through an environment will revisit most objects. Performances drop when the object is not seen and increase when the object is recalled. This raises the question: to what extent do periodic recalls of older objects impact model performance, particularly in terms of learning and forgetting? Specifically, to what extent do intermediate recalls in the data stream help compensate for catastrophic forgetting?

A. QUANTIFYING FORGETTING AND EXPERIMENTAL SETUP

As explained in Section II-D, we use the EgoObjects CL Instance benchmark, which consists of five experiences, to evaluate forgetting in the Faster R-CNN. In each experiment, we train a Faster R-CNN [28] sequentially on each experience. The model is built with a Mobile-Net [63] as Backbone, pre-trained on the COCO dataset [54]. The model is trained by stochastic gradient descent with a learning rate of 0.01 and a weight decay of 10^{-5}.

To quantify forgetting, we measure the model's Average Precision at an Intersection over Union of 0.5 (AP50) on the objects in the first experience D_1 as we train the model on subsequent experiences $D_2, D_3, \ldots, D_T$. This approach allows us to observe and measure how the model's ability to recognize objects from D_1 deteriorates as it is exposed to new data in the following experiences.

Figure 2 shows the AP50 of D_1 objects when we train a Faster R-CNN model on each experience of the EgoObjects

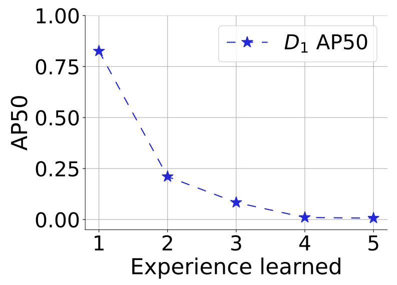


FIGURE 2. Performance of the Faster R-CNN model on the objects from the first experience D_1 as it is sequentially trained on each subsequent experience $D_2, D_3, \ldots, D_5$. The decreasing performance curve illustrates how the model gradually forgets the objects from D_1 as it learns new objects from later experiences.

CL Instance benchmark sequentially. The decreasing performance curve shows that the model gradually loose knowledge about objects in D_1 as it learns other objects in later experiences. Indeed, by the end of training on the whole data stream, the model is unable to locate objects from the first experience D_1 as AP50 reaches 0.

In the following, we manually trigger recalls on D_1 and analyze the response of the Faster R-CNN to determine the extent to which it successfully retrieves information about forgotten objects.

B. EVALUATING THE IMPACT OF INTERMEDIATE RECALLS

We conduct four different experiments on the EgoObjects CL Instance [52] by performing recalls on D_1 at different moments in the data stream. A recall involves retraining the model on all data contained in D_1.

- **Experiment (a):** All experiences from D_1 to D_5 are successively learned without any intermediate recalls.
- **Experiment (b):** D_1 is recalled after learning D_2.
- **Experiment (c):** D_1 is recalled after learning both D_2 and D_3.
- **Experiment (d):** D_1 is recalled after learning D_2, D_3, and D_4.

In all four experiments, we perform a final recall at the end of the stream after learning D_5. The AP50 on D_1 objects is evaluated four times per experience, allowing for a better analysis of the AP50 curve. We show the results of the four experiments in Figure 3.

Figure 3 shows that the model gradually forgets knowledge of D_1 when learning other experiences. The performance of objects in D_1 drops when learning on D_2, D_3, D_4 and D_5. During forgetting phases, we found that the performance curves looks like an exponential decay. To quantify this observation, we performed exponential regressions of the form $t \longrightarrow E_0 \exp\left(-\frac{t}{\tau}\right)$, where t is the training iteration step, and (E_0, τ) are the regression parameters. We define τ as the *knowledge retention rate*, indicating how information is retained over time.

We did such a regression on D_1 AP50 after each intermediate recall. We found as intermediate reminders are added, the information retention rate increases : the knowledge retention

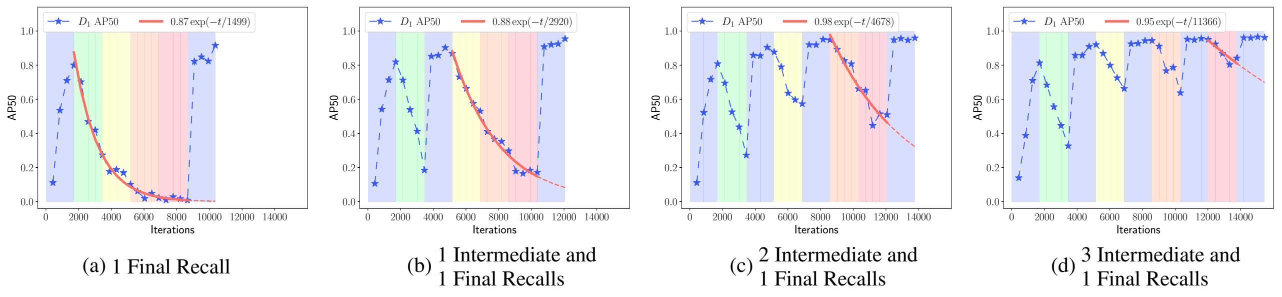


FIGURE 3. **Performance of a Faster R-CNN with periodic recalls on objects from first experience with different numbers of intermediate recalls. Evaluation is performed on objects from the first experience D_1 in the EgoObjects dataset, measured by AP50 (Average Precision at 50% IoU). The colored bands represent the times when the model was trained on a specific experience. Blue bands indicate training phases on D_1, while other colors represent training on different experiences with disjoint objects from D_1. When performance decreases due to catastrophic forgetting, it follows an exponential decay of the form $t \longrightarrow E_0 \exp(-t/\tau)$. Values of parameters E_0 and τ are given in each figure. A recall on D_1 rapidly restores the model's performance to its previous level on D_1. Successive recalls improve memory stability, increasing the knowledge retention rate τ from approximately 1,500 iteration steps (without recall) to around 11,370 steps, suggesting that successive recalls enhance knowledge retention.**

rate τ increases from around 1500 iteration steps (Fig. 3(a)) to around 11370 (Fig. 3(d)). This means that without recall, the model will lose 63% of its knowledge after 1500 iterations steps. However, after three intermediate recalls, the model loses only 12% of its knowledge after 1500 iterations, suggesting that the model's knowledge retention improves with more recalls. This implies that periodic recalls can help the model to consolidate its knowledge over time and reduce the performance decay due to forgetting over the long term.

These findings align with the work of Hermann Ebbinghaus, a pioneering psychologist in the late 19th century who made significant contributions to the study of human memory and forgetting. Ebbinghaus's experiments involved memorizing lists of nonsense syllables and then measuring how quickly humans forgot them over time. His findings revealed that memory decay occurs shortly after learning, but the rate of forgetting slows down with time, forming a curve known as the *forgetting curve* depicted in Fig. 4.

Ebbinghaus's findings are relevant to our study as they suggest that periodic recalls can increase the rate of knowledge retention, which we observed in our experiments with periodic recalls in the data stream. His research demonstrated that forgetting curves follow an exponential decay pattern. Our results indicate that the model's performance can recover through recalls, mirroring Ebbinghaus's observation that revisiting information can help maintain memory.

Additionally, we found that each recall on D_1 restores performance to the level achieved during the last training phase on this same experience. This is true even in scenarios with long forgetting spans. For example, in the first experiment presented in Figure 3(a), the model is trained sequentially on D_2 through D_5, where its performance on D_1 drops to 0 AP50. During the final recall on D_1, the model reaches its peak performance within a few iteration steps. This suggests that the model is able to quickly recover performance on old objects even if it has forgotten how to recognize them.

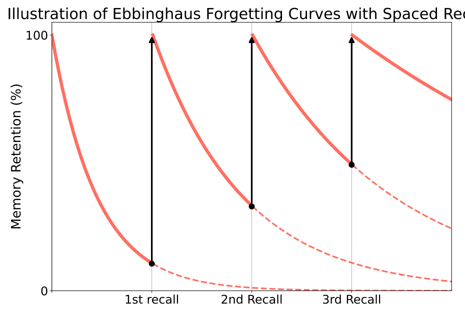


FIGURE 4. **Ebbinghaus Forgetting Curves with Intermediate Recalls. This plot illustrates the decay of memory retention over time based on the Ebbinghaus forgetting curve model. Five separate curves represent memory retention after different intervals of recall. The decay rate decreases with each subsequent recall, demonstrating the effect of repeated reviews on memory retention. We observed similar forgetting curves in the performance of the artificial neural network architecture Faster R-CNN when trained sequentially on a sequence of experiences.**

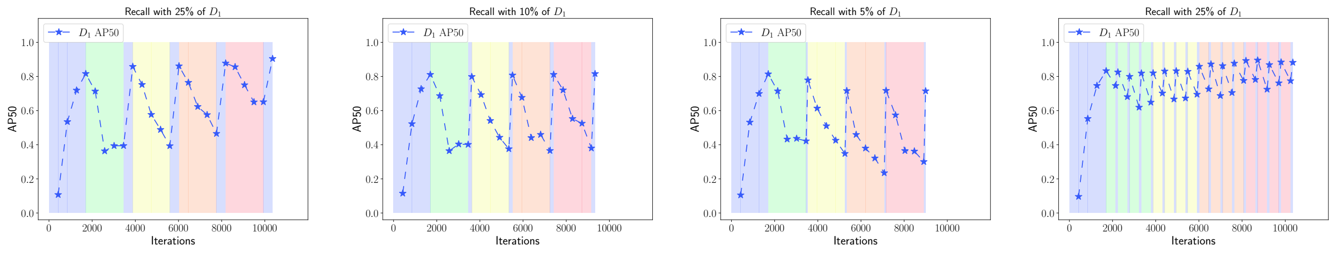
C. OPTIMIZING RECALL SIZE FOR EFFICIENT PERFORMANCE RECOVERY

In previous experiments, we performed recalls on old data by replaying the whole D_1. However, our results suggest that only a few iteration steps are sufficient to retrieve performance. This suggests that recalls could be optimized by decreasing their size, raising the question: "How much data do we need to retrieve performance during a recall?"

To answer this question, we performed three experiments, where at each recall, only a subset of D_1 was used, with sizes of 25%, 10%, and 5% of the original dataset. The results are shown in Fig. 5.

Our findings in Figure 5(a) indicate that recalling 25% of the original D_1 dataset is sufficient to improve performance after each intermediate recall. For the 10% subset in Figure 5(b), performance remains stable. However, Figure 5(c) shows that recalling only 5% of the D_1 dataset is insufficient, as the performance declines over time.

These findings suggest that the frequency and amount of recall are important for ensuring good performance in the long term and counterbalancing the catastrophic forgetting



(a) Recalls with 25% of D_1 (b) Recalls with 10% of D_1 (c) Recalls with 5% of D_1 (d) High frequency recalls (6.25% of D_1 every 400 iterations)

FIGURE 5. Impact of varying the size and frequency of recalls on Faster R-CNN performance over time. The evaluation is performed on objects from the first experience (D_1) in the EgoObjects dataset, measured by AP50 (Average Precision at 50% IoU). This figure shows the results of experiments with different recall sizes (25%, 10%, and 5% of the original D_1 dataset). The plots illustrate how these different recall sizes affect the model's ability to retain and regain performance after a recall. Recalling 25% of the dataset leads to improved performance, recalling 10% results in stable performance, while recalling only 5% results in a decline. Additionally, distributing a 25% recall into 4 intermediate recalls of 6.25% achieves similar performance improvements. Increasing the frequency of small recalls is effective for mitigating catastrophic forgetting.

phenomenon. In our setup, recalling 5% of the D_1 dataset seems insufficient, but the frequency of recall is low with only four recalls for four later experiments learned. In real-world scenarios, small and frequent recalls of objects can happen.

Fig. 5(d) shows an experiment with frequent and small recalls. In this setup, we split 25% of D_1 into four recalls, leading to a recall of 6.25% of D_1 every 400 iterations rather than every 1600 iterations as in previous experiments. We observed that this setup with frequent and small recalls achieves the same final performance as the setup with larger and less frequent recalls. When old data is frequently recalled, the model performs better on average over time. These findings highlight the importance of recalls for maintaining performance. Recalls help to quickly recover lost knowledge and reduce the forgetting rate.

Finally, we observed rapid performance recovery during recalls, indicating that some knowledge is still preserved despite the observed forgetting decay. This suggests that the model retains some information in its weights about previously learned objects. In other terms, forgetting is exacerbated in some parts of the architecture. This leads to the following question: Where is localized the forgetting in the model architecture ? Understanding these internal mechanisms is key to enhancing the model's resilience to forgetting in dynamic learning environments. In the next section, we propose a new methodology to investigate where forgetting occurs within the Faster R-CNN architecture.

IV. LOCATING FORGETTING IN THE FASTER R-CNN ARCHITECTURE

To mitigate catastrophic forgetting in the Faster R-CNN architecture, it is necessary to identify the most responsible modules. This allow to further propose remediation methods that focus on them. In this section, we introduce our module probing methodology, which locate where the forgetting occurs in the Faster R-CNN architecture. After presenting the methodology and experimental setting, we present and comment on the obtained results.

A. MODULE PROBING PROTOCOL

Consider a Faster R-CNN initially trained on a dataset D_1, and then further trained on a new dataset D_2. During this second phase of training, the model gradually loses its ability to recognize objects from D_1, resulting in a noticeable decline in performance on the original dataset. As the model adapts to the new data, its weights shift to a part of the weight space optimized for recognizing objects from D_2. This shift demonstrates a phenomenon known as catastrophic forgetting [13], [41], where previously learned information is overshadowed by new learning. Fig. 6 illustrates how this forgetting occurs over the model's training phases.

The proposed Module Probing methodology assesses the forgetting in the Faster R-CNN across various depths in its architecture. This methodology systematically evaluates the extent to which the model retains its ability to recognize objects from D_1 after being retrained on D_2. By focusing on a specific set of weight parameters, θ_{test}, we can probe their position in the weight space. This helps us determine whether these parameters are still effective in recognizing D_1 objects or if they have entirely shifted to recognizing D_2 objects. By analyzing the results of this probing, we gain insights into the model's retention capabilities and the extent of catastrophic

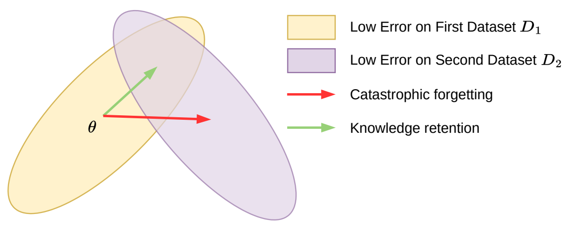

Low Error on First Dataset D_1
Low Error on Second Dataset D_2
Catastrophic forgetting
Knowledge retention
θ

FIGURE 6. Illustration of the catastrophic forgetting effect for a model of parameters θ. When the network is trained on a new dataset D_2, gradient updates may damage the performance on the previous dataset D_1. The model efficiently maintains knowledge if it finds a location in weight space suitable for recognizing both new and old objects.

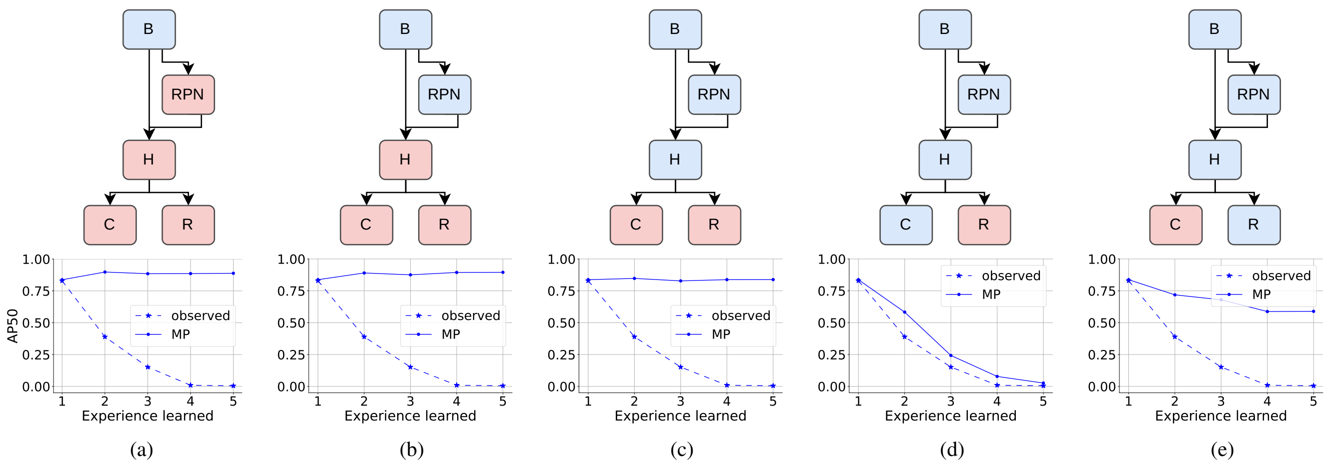


FIGURE 7. Performance comparison of *Observed* and *Module Probed (MP)* performances across five Module Probing configurations (a) - (e). The Faster R-CNN is composed of five modules : a Backbone (B), a RPN, a Linear Head (H), a Box Classifier (C) and a Box Regressor (R). Modules marked in blue are frozen to examine their knowledge retention. When the Box Classifier is excluded from retraining, the model struggles to readjust its performance. In all other configurations, the model successfully readjusts. These observations strongly suggest that the Softmax layer is the main source of forgetting within the architecture.

forgetting.

The Module Probing methodology involves the following steps:

1) **Initialization:** Train the model on D_1
2) **Sequential Training:** Train the model on D_2. The model faces forgetting of objects from D_1.
3) **Partial Freezing:** After training on D_2, freeze a subset of the model's parameters to probe (θ_{test}).
4) **Recall:** Train the partially frozen model on D_1 again. At the end of this recall, θ_{test} parameters are still in their post-D_2 training as they were frozen. The rest of the model parameters are fine-tuned to recognize objects from D_1.
5) **Evaluation:** Measure the model's performance on D_1 to determine if the frozen modules have retained information from the initial training. This performance is called the *Module Probed* performance.

If the model shows low performance on D_1 after the recall, it indicates that θ_{test} did not retain the necessary information to achieve the previous performance levels, even with the recall phase for the rest of the model. This suggests that the information has been lost in the frozen modules. Conversely, if the model regains the lost performance from the initial training on D_1, it indicates that knowledge about D_1 has been retained in θ_{test}, despite the training on D_2.

B. TRAINING SETUP

In our Module Probing experiments, we use the EgoObjects CL Instance benchmark which comprises five experiences D_1 to D_5. This NR-free benchmark ensures that objects in D_1 are not present in subsequent experiences. The Module Probing methodology is described for a single experience D_2 only. In our experiments, we repeat the steps 2) to 5) of the methodology for the other experiences D_3, D_4 and D_5. This leads to four Module Probed performance values and allow for a longer analysis.

The same experimental setup is used as in section III. Specifically, we train a Faster R-CNN [28] with a MobileNet [63] pre-trained on the COCO dataset [54]. The model is trained by stochastic gradient descent with a learning rate of 0.01 and a weight decay of 10^{-5}. Module Probing evaluation is done with the AP50 on objects from D_1.

We employ a top-down approach to measure forgetting in each module of the Faster R-CNN. We perform a Module Probing experiment by freezing only the Backbone. Then, we perform additional experiments in which we progressively freeze following modules. This results in five different freezing configurations, illustrated at the top of Figure 7.

For each freeze configuration, we plot two curves in Figure 7. In addition to the Module Probed performances, we show the *observed* performance, which is the forgetting curve of the model when trained sequentially through D_1 to D_5 without recalls and without freeze.

C. RESULTS

As depicted in section III, the Faster R-CNN loses knowledge of previous data: the *observed* performance curve decreases as an exponential decay. This performance is the same in all five experiments in Figure 7 as it does not depend on the freeze configuration of the Module Probing methodology.

In the first three configurations (7(a), 7(b), 7(c)), the *MP* AP50 is readjusted compared to *observed*. This suggests that the frozen modules have retained information relating to the first experience D_1. Thus, the forgetting effect in the Faster R-CNN is not localized in the first three modules Backbone, RPN, and Linear Head.

In the fourth freeze configuration where the Box Classifier is additionally frozen (7(d)), the performance *MP* is not readjusted: the *MP* and *observed* curves are almost overlapping. In this configuration, only the Box Regressor is re-trained on D_1, the rest of the architecture is frozen. Interestingly, this is the only configuration among five where the Box Classifier is frozen.

In the last configuration (7(e)), the Box Classifier is the only module retrained on D_1. In this configuration, the model performance is readjusted. This indicates that all the remained frozen modules, i.e. all modules except the Box Classifier, forget limited knowledge linked to the first experience D_1.

D. DISCUSSION

These results suggest that the Box Classifier module, which is a Softmax classification layer, is primarily responsible for the overall forgetting effect in the Faster R-CNN architecture.

In OCL for classification problems [3], the Softmax layer has been criticized in numerous works [3], [64], [65]. This layer induces a classification bias towards more recent classes, leading to poor classification performance, despite limited intermediate representation forgetting [66].

To address the issue of forgetting in OOD within the Faster R-CNN architecture, it is essential to focus on the Box Classifier module, as this is where forgetting is most pronounced. In the following section, we propose and evaluate various methods to compensate for this forgetting. Specifically, we introduce an enhancement to the Box Classifier designed to mitigate the effects of forgetting and improve overall model performance.

V. FORGETTING COMPENSATION IN FASTER R-CNN FOR OOD

In this section, we introduce new method called Configurable Recall (CR) to counteract the catastrophic forgetting effect in an OOD learning setting. This method allow a Faster R-CNN to have a stable progressive learning when trained on a sequence of experiences. CR incorporates two elements based on empirical results obtained in sections III and IV. Firstly, it uses an external memory buffer to store old data samples and perform periodic recalls with it. Secondly, we modify the Faster R-CNN architecture at the softmax layer level in order to mitigate forgetting. We detail our method and demonstrate its performance on three OOD benchmarks.

A. CONFIGURABLE RECALL

1) Learning with recalls

In section III, we examined the effects of periodic recalls on model performance. Our experiments demonstrated that intermediate recalls can significantly mitigate catastrophic forgetting. This suggests that integrating regular recalls into the training process can help maintain model performance over time, reducing the impact of forgetting. We note that NR, present in many OOD scenarios, has this recall effect but we cannot control it.

To alleviate this problem, we propose to perform recalls with old data stored in an external memory. Using an external memory buffer for OCL is a common strategy to counteract catastrophic forgetting. But, in the OOD context, it is yet unclear how much data should be replayed, and at what frequency, to get the better performances. We note CR_p^l the CR method which performs recalls on old data with period p and recall length l. Every p iterations, the CR_p^l method stops training on the stream and perform an offline recall of a size l iterations with images randomly sampled from the memory buffer. After recall, the model continue its training on the data stream. The memory can contain up to M images. The memory buffer is implemented with a Reservoir Sampling strategy [67] which is common used in OCL [3].

2) Mitigating forgetting with Balanced loss

In the previous section, our Module Probing experiences showed that catastrophic forgetting in Faster R-CNN is mainly localized in the Box Classifier module which is a Softmax classification layer.

When trained in a context of class imbalance, the Softmax classifier is subject to a bias towards over-represented classes [68]. In OOD, recent classes tend to be more represented than old classes. As a result, Softmax favors new classes at the expense of old ones, which is perceived as knowledge forgetting [3], [65].

To overcome the problem of Softmax bias in Faster R-CNN, we replace the original cross-entropy loss for classification with the Balanced loss [3] defined as follows:

$$L_{Balanced}(p_i, y_i) = -\log\left(\frac{e^{z_{y_i}}}{\sum_{j\in C}\delta_j e^{z_j}}\right), \delta_j = \begin{cases} 0 \text{ if } j \notin C_{\mathrm{cur}} \\ 1 \text{ if } j \in C_{\mathrm{cur}} \end{cases} \tag{3}$$

where p_i is the region of interest i of ground truth class y_i. z_j is the logit associated with the region of interest p_i for class j. C is the set of all classes while C_{cur} is the set of classes currently present in the minibatch. By exclusively considering outputs corresponding to classes in the current minibatch, the training avoids excessively penalizing logits for old classes.

B. EXPERIMENTAL SETTING

1) Benchmarks

We evaluated CR_p^l on three benchmarks for OOD presented in Section II: OAK [24], EgoObjects CL Instance and EgoObjects CL Category [52]. Each benchmark scenario is composed of five experiences. For the two EgoObjects benchmarks, we use the original scenario decomposition [52]. For OAK [24], we divide the entire data stream into five segments, ensuring that each experience contains a similar number of images.

We use the methodology established by recent works in OOD for the allocation of training and test data [24], [26], [27] on OAK and EgoObjects datasets. Every 16 consecutive video frames, we reserve one frame for testing while the remaining frames are used for training. We perform one evaluation step after each training experience on each benchmark.

The three benchmarks show different levels of Natural Replay. We show in Table 1 the Natural Replay Score (NRS) [27] for each benchmark. NRS ranges between 0 and 1 and the closer the NRS is to 1, the more often classes are revisited in the stream i.e. the Natural Replay is higher. The three benchmarks provide a broad spectrum of NRS values,

TABLE 1. Comparison of Final Average Precision (FAP), Continual Average Precision (CAP), and Training Time (T) in minutes for various strategies (Incremental, GDumb, ER, Efficient-CLS and CR) using Softmax or Balanced Loss across three benchmarks (EgoObjects CL Instance, EgoObjects CL Category, OAK). Higher Natural Replay Scores (NRS 2) indicate more class revisits in the scenario. ↑ indicates higher is better. ↓ indicates lower is better. The training batch composition is indicated for each strategy: only stream, only memory, a combination of both, or exclusively stream or memory.

Training Batch Composition		EgoObjects CL Instance ($NRS = 0$)			EgoObjects CL Category ($NRS = 0.51$)			OAK ($NRS = 0.92$)		
		FAP (↑)	CAP (↑)	Time (↓)	FAP (↑)	CAP (↑)	Time (↓)	FAP (↑)	CAP (↑)	Time (↓)
Stream	Offline	73.1	-	27.6	67.9	-	29.1	28.8	-	13.6
	Incremental	27.3	25.6	27.4	42.2	39.5	28.6	27.8	21.4	13.9
Memory	GDumb (1 epoch)	36.6	22.0	11.0	47.3	31.8	11.2	27.7	21.7	11.8
	GDumb (5 epochs)	80.9	50.6	49.1	73.5	54.0	50.8	41.1	34.1	52.4
Stream AND Memory	ER [15]	83.5	50.5	52.3	69.3	50.5	50.9	30.3	22.3	25.4
	Efficient-CLS [26]	85.6	51.0	48.7	70.8	51.6	50.3	30.2	23.6	24.0
Stream XOR Memory	CR_{1000}^{50}	34.7	30.7	33.9	58.7	45.1	34.5	30.2	23.7	16.8
	w/ Balanced	66.7	42.8	33.2	66.3	50.4	34.5	31.8	24.9	16.5
	CR_{1000}^{500}	48.4	38.6	61.9	71.5	53.4	64.4	32.5	25.7	26.3
	w/ Balanced	84.4	50.4	61.2	73.4	54.7	65.3	33.8	26.9	25.3
	CR_{100}^{50}	74.4	47.7	60.6	70.4	52.4	64.5	32.4	25.4	30.1
	w/ Balanced	86.9	51.4	61.6	73.6	54.7	68.2	34.1	26.5	27.3

allowing the comparison of methods in scenarios with varying frequencies of old object revisits. This helps in evaluating how well different methods perform when objects are revisited either less frequently or more often.

2) Evaluation Metrics

We use three evaluation metrics [24]. Let CAP_t denotes the AP at IoU 0.5 after training on D_t and T is the total number of experiences.

Final Average Precision (FAP): FAP assesses the model's overall performance at the end of training:

$$FAP = CAP_T \tag{4}$$

Continual Average Precision (CAP): CAP evaluates the model's continual learning performance throughout training as the AP average over time:

$$CAP = \frac{1}{T}\sum_{t=1}^{T} CAP_t \tag{5}$$

Training Time (Time): This metric measures the total duration required to train the model on the scenario, including any intermediate recalls. It evaluates the efficiency and feasibility of compared training strategies.

3) Comparison Baselines

We compare our method CR to the following baselines:

Offline: The Faster R-CNN is trained jointly on all experiences in an offline manner for one epoch. This serves as reference representing a Faster R-CNN trained on a scenario where no catastrophic forgetting occurs.

Incremental: The Faster R-CNN is trained over the data stream without any precautions to prevent catastrophic forgetting. This serves as a lower bound, where catastrophic forgetting is maximal.

Gdumb [39]: This method trains the model only on images from memory, rather than directly on the stream. The memory buffer is balanced across all seen objects to ensure fair representation. We present the results of this method with one epoch and five training epochs per experience.

ER (Experience Replay) [15]: This method augments the current batch with old images stored in an external memory buffer. ER enables training on both new data from the stream and old data to preserve knowledge, and it has proven effective in OCL [3] as well as in OOD contexts [27].

Efficient-CLS [26]: Originally proposed for semi-supervised OOD, this method uses two complementary learning systems: a fast learner that quickly adapts to new data and a slow learner that consolidates knowledge over time. The slow learner generates pseudo-labels for unlabeled video frames, guiding the fast learner and reducing annotation costs while minimizing forgetting. The fast learner is implemented as the ER method, whereas the slow learner is trained as an exponential moving average of the fast learner. We compare our method to Efficient-CLS in our context which is fully supervised, i.e. where all images are annotated.

In our experiments, each method uses a Faster R-CNN [28] with a Mobile-Net backbone [63], initialized with weights trained on the COCO dataset [54]. We use Stochastic Gradient Descent with a learning rate of 0.01 and weight decay of 10^{-5}, the batch size is set to 8. For ER and Efficient-CLS, each batch of 8 images is augmented with 8 images randomly sampled from memory, forming a training batch of 16 images.

C. RESULTS

Table 1 presents the results comparing the aforementioned methods to ours CR. We categorize all compared methods according to whether the training is conducted exclusively on the data stream, exclusively on memory, or on a combination of both.

1) Stream against Memory Based Methods

Our results show that memory-based techniques perform better. This can be explained for two key reasons. Firstly, replaying old data effectively mitigates forgetting, as shown in the previous section III, which demonstrated that recalling old data helps maintain performance over time. Secondly, it allows multiple passes on the same data. Data is seen only once in the stream for Offline and Incremental baselines. Whereas, the same data is revisited through the memory buffer in the memory based methods. This dual exposure ensures that memory-based methods often outperform the offline baseline, which trains on all data in a single pass without the benefit of reinforcement from repeated exposures. For example, GDumb's performance improves significantly when the number of training epochs is increased. Specifically, on the EgoObjects CL Instance benchmark, FAP increase from 36.6 to 80.9 and CAP increases from 22 to 50.6.

2) Configurable Recall

We present the results for CR_{1000}^{50}, CR_{1000}^{500} and CR_{100}^{50}. On the EgoObjects CL Instance benchmark, our results reflects our finding from section III. Firstly, performing recalls with a larger proportion of data lead to better performances: CR_{1000}^{500} outperforms CR_{1000}^{50} by 13.7 FAP points and 7.9 CAP points. Secondly, frequent and small recalls lead to better results rather than important and less frequent recalls: CR_{100}^{50} outperforms CR_{1000}^{500} by 26 FAP points and 9.1 CAP points. Adding the Balanced loss in CR consistently improves performance, particularly in EgoObjects CL Instance which has no NR. Additionally, our Balanced loss operates without additional computational costs, maintaining the same training time as the Softmax layer.

Finally, CR_{100}^{50} with Balanced loss improves performance to attain 86.9 FAP and 51.4 CAP points on EgoObjects CL Instance. It also achieves 73.6 FAP and 54.7 CAP on EgoObjects CL Category. Our method CR_{100}^{50} outperforms all other existing methods on both EgoObjects benchmarks. On OAK, CR_{100}^{50} achieves great results with 34.1 FAP and 26.5 CAP. This is lower than GDumb with five epochs, but the latest needed around 2 times more training time. Given these substantial benefits, our strategy mixing frequent recalls and the Balanced loss is a highly effective strategy for OOD.

D. PERFORMANCE VARIATIONS ACROSS DIFFERENT NR LEVELS

Our results indicate that performance gaps between strategies are more pronounced in low NR benchmarks like EgoObjects CL Instance than in high NR benchmarks like OAK. In low NR scenarios, forgetting is more severe, making methods like CR, which address forgetting, highly effective compared to strategies like Incremental, which are prone to forgetting. In high NR scenarios, the frequent reappearance of objects already mitigates forgetting: the model's retention of knowledge is not due to an inherent resistance to catastrophic forgetting, but rather to the reappearance of the object in question. In these high NR scenarios, our method allows for selecting shorter sizes and longer periods of recall, making it more efficient than other replay methods. For instance, ER and Efficient-CLS perform recalls at every batch, which is unnecessary in scenarios with an NRS of 0.92 where old data is replayed frequently. These observations highlight the necessity of evaluating OOD methods across a spectrum of NR conditions.

Additionally, our results show that the Balanced loss consistently improves performance across various benchmarks, particularly in scenarios where the NRS is 0, such as EgoObjects CL Instance. Even in scenarios with higher NRS, our loss still significantly enhances performance, and it operates without additional computational costs, maintaining the same training time as the Softmax layer. Given these substantial benefits, the Balanced loss is a highly effective strategy, making it a compelling choice for improving model performance in OOD scenarios across all NRS levels.

VI. CONCLUSION AND FUTURE DIRECTIONS

In this study, we have explored how the Faster R-CNN model acquires and forgets knowledge in an OOD scenario. Our research made the following three main contributions to the field of OOD.

1) **The Faster R-CNN forgets knowledge in a similar manner to humans. Frequent recalls help maintain knowledge:** By analyzing the effects of intermediate recalls of old data on performance, we discovered that the learning and forgetting dynamics of the Faster R-CNN mirror human cognitive processes. Knowledge loss occurs exponentially over time, and successive recalls significantly improve knowledge retention. Additionally, our empirical findings highlight the importance of both the frequency and length of recalls in effectively relearning objects. These insights underscore the value of incorporating regular recall mechanisms to maintain model performance over extended periods.
2) **Forgetting in the Faster R-CNN is mostly localized in the final Softmax classification layer:** We introduced the Module Probing methodology which serves to analyze and quantify the extent of forgetting across different components of the Faster R-CNN architecture. Our results revealed that forgetting is mainly localized in the Softmax classification layer. This finding highlights the importance of focusing on the classification layer when developing strategies to mitigate catastrophic forgetting for the Faster R-CNN in an OOD scenario.
3) **Our proposed strategy Adaptive Recall (AR) and our Balanced loss efficiently mitigate forgetting:** Our study showed that our CR strategy significantly improve model performance in OOD scenarios. CR demonstrates substantial performance gains by using frequent, short recalls to stabilize the learning process and mitigate forgetting. The Balanced loss incorporated in CR enhances performances, particularly in NR-free scenarios, by mitigating the forgetting occurring in the

classification layer. Our strategy is especially effective in low NR scenarios, where forgetting is more severe.

Looking ahead, our work opens several promising areas for future research in the field of OOD. Firstly, optimizing the parameters of periodic recalls, such as their frequency and length, could further enhance their efficiency and effectiveness. This includes investigating the optimal trade-off between recall frequency and training time to maximize model performance while minimizing computational costs. Secondly, exploring alternative architectural modifications of the classification layer may yield new insights into mitigating catastrophic forgetting. Thirdly, developing more sophisticated benchmarks that include varying levels of NR is essential for accurately evaluating and improving OOD models. New benchmarks could ensure comprehensive testing of a model on the same data but with different NR. These directions could lead to advances in the theoretical understanding of catastrophic forgetting in the OOD context and drive improvements in intelligent systems operating in dynamic environments.

ACKNOWLEDGMENT

All the computations presented in this paper were performed using the GRICAD infrastructure (https://gricad.univ-grenoble-alpes.fr), which is supported by Grenoble research communities.

REFERENCES

[1] U. Hasson, S. A. Nastase, and A. Goldstein, "Direct fit to nature: an evolutionary perspective on biological and artificial neural networks," *Neuron*, vol. 105, no. 3, pp. 416–434, 2020.

[2] D. Kudithipudi, M. Aguilar-Simon, J. Babb, M. Bazhenov, D. Blackiston, J. Bongard, A. P. Brna, S. Chakravarthi Raja, N. Cheney, J. Clune, *et al.*, "Biological underpinnings for lifelong learning machines," *Nature Machine Intelligence*, vol. 4, no. 3, pp. 196–210, 2022.

[3] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, "Online Continual Learning in Image Classification: An Empirical Survey," *arXiv:2101.10423 [cs]*, Oct. 2021. arXiv: 2101.10423.

[4] J. He, R. Mao, Z. Shao, and F. Zhu, "Incremental Learning In Online Scenario," *arXiv:2003.13191 [cs]*, Apr. 2021. arXiv: 2003.13191.

[5] R. Aljundi, K. Kelchtermans, and T. Tuytelaars, "Task-Free Continual Learning," *arXiv:1812.03596 [cs, stat]*, Aug. 2019. arXiv: 1812.03596.

[6] T. Sun, M. Segu, J. Postels, Y. Wang, L. Van Gool, B. Schiele, F. Tombari, and F. Yu, "Shift: a synthetic driving dataset for continuous multi-task domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21371–21382, 2022.

[7] E. Verwimp, K. Yang, S. Parisot, L. Hong, S. McDonagh, E. Pérez-Pellitero, M. De Lange, and T. Tuytelaars, "Clad: A realistic continual learning benchmark for autonomous driving," *Neural Networks*, vol. 161, pp. 659–669, 2023.

[8] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges," *arXiv:1907.00182 [cs]*, Nov. 2019. arXiv: 1907.00182.

[9] Q. She, F. Feng, X. Hao, Q. Yang, C. Lan, V. Lomonaco, X. Shi, Z. Wang, Y. Guo, Y. Zhang, *et al.*, "Openloris-object: A robotic vision dataset and benchmark for lifelong deep learning," in *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 4767–4773, IEEE, 2020.

[10] D. Bohus, S. Andrist, A. Feniello, N. Saw, and E. Horvitz, "Continual learning about objects in the wild: An interactive approach," in *Proceedings of the 2022 International Conference on Multimodal Interaction*, pp. 476–486, 2022.

[11] K. Doshi and Y. Yilmaz, "Continual learning for anomaly detection in surveillance videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[12] B. Kwon and T. Kim, "Toward an online continual learning architecture for intrusion detection of video surveillance," *IEEE Access*, vol. 10, pp. 89732–89744, 2022.

[13] M. McCloskey and N. J. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem," vol. 24 of *Psychology of Learning and Motivation*, pp. 109–165, Academic Press, 1989. ISSN: 0079-7421.

[14] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[15] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. S. Torr, and M. Ranzato, "On Tiny Episodic Memories in Continual Learning," *arXiv:1902.10486 [cs, stat]*, June 2019. arXiv: 1902.10486.

[16] D. Lopez-Paz and M. Ranzato, "Gradient Episodic Memory for Continual Learning," *arXiv:1706.08840 [cs]*, Nov. 2017. arXiv: 1706.08840.

[17] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," in *International Conference on Learning Representations*.

[18] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental Classifier and Representation Learning," *arXiv:1611.07725 [cs, stat]*, Apr. 2017. arXiv: 1611.07725.

[19] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars, "Online Continual Learning with Maximally Interfered Retrieval," *arXiv:1908.04742 [cs, stat]*, Oct. 2019. arXiv: 1908.04742.

[20] Y. Guo, B. Liu, and D. Zhao, "Online continual learning through mutual information maximization," in *Proceedings of the 39th International Conference on Machine Learning* (K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, eds.), vol. 162 of *Proceedings of Machine Learning Research*, pp. 8109–8126, PMLR, 17–23 Jul 2022.

[21] Z. Cai, O. Sener, and V. Koltun, "Online continual learning with natural distribution shifts: An empirical study with visual data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8281–8290, October 2021.

[22] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: a strong, simple baseline," *Advances in neural information processing systems*, vol. 33, pp. 15920–15930, 2020.

[23] W. Baptiste, P. Denis, O. Serge, and H. Sylvain, "Online class incremental learning with one-vs-all classifiers for resource constrained devices," in *2023 International Symposium on Image and Signal Processing and Analysis (ISPA)*, pp. 1–6, IEEE, 2023.

[24] J. Wang, X. Wang, Y. Shang-Guan, and A. Gupta, "Wanderlust: Online continual object detection in the real world," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10829–10838, 2021.

[25] M. Acharya, T. Hayes, and C. Kanan, "Rodeo: Replay for online object detection," in *British Machine Vision Conference*, 2020.

[26] J. Z. Wu, D. J. Zhang, W. Hsu, M. Zhang, and M. Z. Shou, "Label-efficient online continual object detection in streaming video," *arXiv preprint arXiv:2206.00309*, 2022.

[27] B. Wagner, D. Pellerin, and S. Huet, "Comparative study of natural replay and experience replay in online object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 3585–3593, October 2023.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[29] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *Proceedings of the IEEE international conference on computer vision*, pp. 3400–3409, 2017.

[30] A. G. Menezes, G. de Moura, C. Alves, and A. C. de Carvalho, "Continual object detection: A review of definitions, strategies, and challenges," *Neural Networks*, 2023.

[31] K. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian, "Towards open world object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5830–5840, 2021.

[32] H. Ebbinghaus, "Memory: A contribution to experimental psychology," *Annals of neurosciences*, vol. 20, no. 4, p. 155, 2013.

[33] A. G. Menezes, G. de Moura, C. Alves, and A. C. de Carvalho, "Continual object detection: a review of definitions, strategies, and challenges," *Neural Networks*, 2023.

[34] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural networks*, vol. 113, pp. 54–71, 2019.

[35] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. arXiv: 1909.08383.

[36] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: theory, method and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[37] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," *Advances in neural information processing systems*, vol. 32, 2019.

[38] D. Shim, Z. Mai, J. Jeong, S. Sanner, H. Kim, and J. Jang, "Online class-incremental continual learning with adversarial shapley value," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, pp. 9630–9638, 2021.

[39] A. Prabhu, P. H. Torr, and P. K. Dokania, "Gdumb: A simple approach that questions our progress in continual learning," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 524–540, Springer, 2020.

[40] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[41] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[42] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International conference on machine learning*, pp. 3987–3995, PMLR, 2017.

[43] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory Aware Synapses: Learning what (not) to forget," *arXiv:1711.09601 [cs, stat]*, Oct. 2018. arXiv: 1711.09601.

[44] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018.

[45] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[46] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3366–3375, 2017.

[47] J. Rajasegaran, M. Hayat, S. H. Khan, F. S. Khan, and L. Shao, "Random path selection for continual learning," *Advances in neural information processing systems*, vol. 32, 2019.

[48] A. Soutif-Cormerais, A. Carta, A. Cossu, J. Hurtado, V. Lomonaco, J. Van de Weijer, and H. Hemati, "A comprehensive empirical evaluation on online continual learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3518–3528, 2023.

[49] Z. Mai, R. Li, H. Kim, and S. Sanner, "Supervised Contrastive Replay: Revisiting the Nearest Class Mean Classifier in Online Class-Incremental Continual Learning," *arXiv:2103.13885 [cs]*, Sept. 2021. arXiv: 2103.13885.

[50] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[51] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 21–37, Springer, 2016.

[52] C. Zhu, F. Xiao, A. Alvarado, Y. Babaei, J. Hu, H. El-Mohri, S. Culatana, R. Sumbaly, and Z. Yan, "Egoobjects: A large-scale egocentric dataset for fine-grained object understanding," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20110–20120, 2023.

[53] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, pp. 303–338, 2010.

[54] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755, Springer, 2014.

[55] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *Proceedings of the IEEE international conference on computer vision*, pp. 3400–3409, 2017.

[56] X. Liu, H. Yang, A. Ravichandran, R. Bhotika, and S. Soatto, "Multi-task incremental learning for object detection," *arXiv preprint arXiv:2002.05347*, 2020.

[57] W. Zhou, S. Chang, N. Sosa, H. Hamann, and D. Cox, "Lifelong object detection," *arXiv preprint arXiv:2009.01129*, 2020.

[58] C. Peng, K. Zhao, and B. C. Lovell, "Faster ilod: Incremental learning for object detectors based on faster rcnn," *Pattern recognition letters*, vol. 140, pp. 109–115, 2020.

[59] A. Cossu, G. Graffieti, L. Pellegrini, D. Maltoni, D. Bacciu, A. Carta, and V. Lomonaco, "Is class-incremental enough for continual learning?," *Frontiers in Artificial Intelligence*, vol. 5, p. 829842, 2022.

[60] H. Hemati, A. Cossu, A. Carta, J. Hurtado, L. Pellegrini, D. Bacciu, V. Lomonaco, and D. Borth, "Class-incremental learning with repetition," *arXiv preprint arXiv:2301.11396*, 2023.

[61] T. Lesort, O. Ostapenko, D. Misra, M. R. Arefin, P. Rodríguez, L. Charlin, and I. Rish, "Scaling the number of tasks in continual learning," *arXiv preprint arXiv:2207.04543*, 2022.

[62] J. T. Walker, *Statistics in criminal justice: Analysis and interpretation*. Jones & Bartlett Learning, 1999.

[63] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[64] H. Ahn, J. Kwak, S. Lim, H. Bang, H. Kim, and T. Moon, "Ss-il: Separated softmax for incremental learning," in *Proceedings of the IEEE/CVF International conference on computer vision*, pp. 844–853, 2021.

[65] G. Liang, Z. Chen, Z. Chen, S. Ji, and Y. Zhang, "New insights on relieving task-recency bias for online class incremental learning," *arXiv preprint arXiv:2302.08243*, 2023.

[66] M. Davari and E. Belilovsky, "Probing representation forgetting in continual learning," in *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

[67] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.

[68] B. Kang, S. Xie, M. Rohrbach, Z. Yan, A. Gordo, J. Feng, and Y. Kalantidis, "Decoupling representation and classifier for long-tailed recognition," *arXiv preprint arXiv:1910.09217*, 2019.