Forgetting Analysis by Module Probing for Online Object Detection with Faster R-CNN

Baptiste Wagner, Denis Pellerin, Sylvain Huet

Forgetting Analysis by Module Probing for Online Object Detection with Faster R-CNN

Baptiste Wagner, Denis Pellerin, Sylvain Huet
Univ. Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab
38000 Grenoble, France

***Abstract*—Online Object Detection (OOD) involves learning novel object categories from a stream of images, like the one generated by an agent exploring new environments. In this scenario, the widely used Faster R-CNN architecture faces catastrophic forgetting—the phenomenon where acquiring new knowledge leads to forget previously learned knowledge. The forgetting evaluations published in the literature focus only on the evolution of the performances on past seen data without questioning where forgetting occurs in the architecture and how it propagates. In this paper, our first contribution introduces a new protocol called Module Probing to offer a detailed evaluation of forgetting. This protocol identifies the layers accountable for catastrophic forgetting within the Faster R-CNN architecture. Our results reveal that forgetting is predominantly concentrated in the final classification layer. Building on these insights, our second contribution involves mitigating forgetting by modifying the architecture's classification layer. We demonstrate that it significantly reduces forgetting on three OOD benchmarks. Our achievements provides a first replay-free baseline for challenging OOD scenarios to enhance model long-term performance.**

***Index Terms*—Catastrophic forgetting, Online Object Detection, Faster R-CNN, Online Continual Learning**

I. Introduction

Online Object Detection (OOD) [1] involves training an object detector on a continuous stream of images. This kind of training system mimics a real-world experience of an embodied agent that encounters different new and already-seen categories of objects as it navigates through.

The reference model architecture for OOD is the Faster R-CNN [2]. However, this architecture is subject to the phenomenon known as catastrophic forgetting [1], [3]–[5]. This phenomenon is characterized by a loss of information related to previously learned data, resulting in an irreversible decline in the model's performances when it is subsequently trained on new data. [1], [5]. Furthermore, the Faster R-CNN architecture is complex and composed of different modules for solving the localization and classification tasks for OOD [2]. The challenge is therefore to identify the components of the architecture responsible for the occurrence of catastrophic forgetting. Indeed, existing metrics for forgetting evaluation only assess the architecture's overall forgetting, without examining the Faster R-CNN components individually.

In this study, we investigate the forgetting manifestation inside the Faster R-CNN architecture trained in an OOD context. To this end, our first contribution is the proposal of a new evaluation protocol called Module Probing. This protocol identifies the architecture modules where forgetting occurs and allow to understand the impact of forgetting at the different layers of the Faster R-CNN architecture.

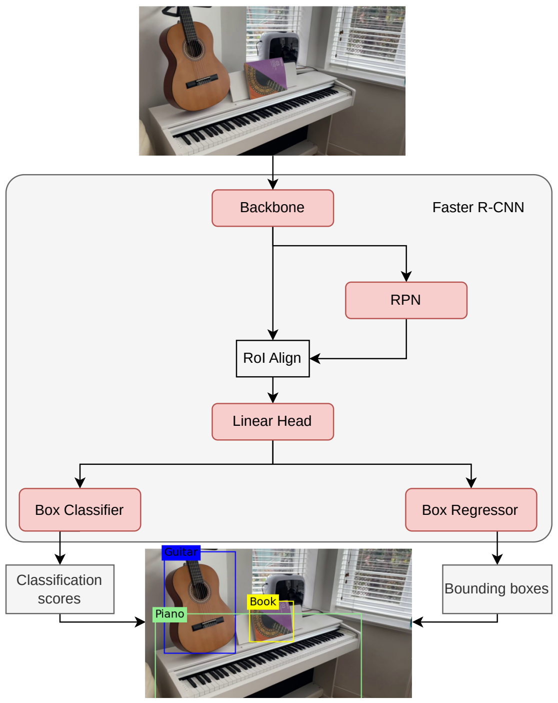


Figure 1. The Faster R-CNN architecture. An input image (here from the EgoObjects benchmark [6]) is fed in a **Backbone** to generate a global image feature map. The **RPN** (Region Proposal Network) identifies regions of interest containing objects. The **RoI Align** operation generates a region-specific feature map for each proposed region. Each RoI Aligned region is fed to two fully connected layers in the **Linear Head**, then to the **Box Classifier** and the **Box Regressor** to output classification scores and refined bounding boxes.

In particular, our analysis with Module Probing identified the last classification layer as the main factor of forgetting in the architecture. Based on this result, our second contribution is a new approach to efficiently mitigate the forgetting of the Faster R-CNN by adjusting its classification layer only.

This paper is organized as follows. In Section II we start with an in-depth description of the Faster R-CNN architecture. In Section III, we detail our Module Probing evaluation protocol aimed at probing forgetting in Faster R-CNN and expose the results obtained. In Section IV, we highlight the effectiveness of our approach to mitigate forgetting in the Faster R-CNN, illustrated by the results on three specific benchmarks for OOD.

II. Preliminaries: Faster R-CNN architecture

Faster R-CNN [2] is a reference architecture for object detection tasks. This architecture is part of the two-stage detectors family: the first stage localizes objects by proposing Regions of Interest (RoI) and the second 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.

III. Module Probing

In the context of OOD, knowledge retention in the Faster R-CNN architecture is essential for a model to be effective over the long term. Our main objective is to identify the modules of the Faster R-CNN architecture where the catastrophic forgetting phenomenon occurs.

In this section, inspired by Linear Probing [7], we present a new protocol called Module Probing. This methodology allows us to evaluate each module of the Faster R-CNN in terms of knowledge preservation. Thanks to this approach, we analyze how each element of the architecture reacts during online learning and we locate areas prone to forgetting.

A. Module Probing Protocol

The proposed Module Probing methodology assesses the Faster R-CNN's information retention across various depths in its architecture.

Let's consider two distinct datasets D_1 and D_2. The Faster R-CNN is initially trained on dataset D_1, and subsequently trained on D_2. During the second training phase on D_2, the Faster R-CNN gradually forgets the knowledge related to D_1. The model's weights move to a place in the weight space where the only concern is recognizing the new objects in D_2. This phenomenon of catastrophic forgetting [8], [9] is illustrated in Figure 2.

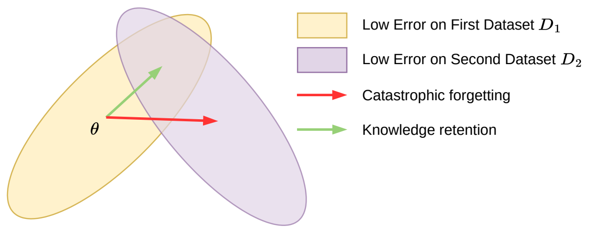

θ
🟨 Low Error on First Dataset D_1
🟪 Low Error on Second Dataset D_2
Catastrophic forgetting
Knowledge retention

Figure 2. 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.

The Module Probing methodology consists of first training the model sequentially on D_1 then on D_2. Secondly, we freeze the modules of interest to examine their information retention. Finally, we retrain the partially frozen model on D_1 and evaluate its performance on the same dataset.

By evaluating on D_1, we measure whether the frozen modules still allow the trained modules to readjust to D_1. If the model has low performance, the frozen modules have forgotten past information. In other terms, their weights are in a place in the weight space exclusive to the resolution of D_2.

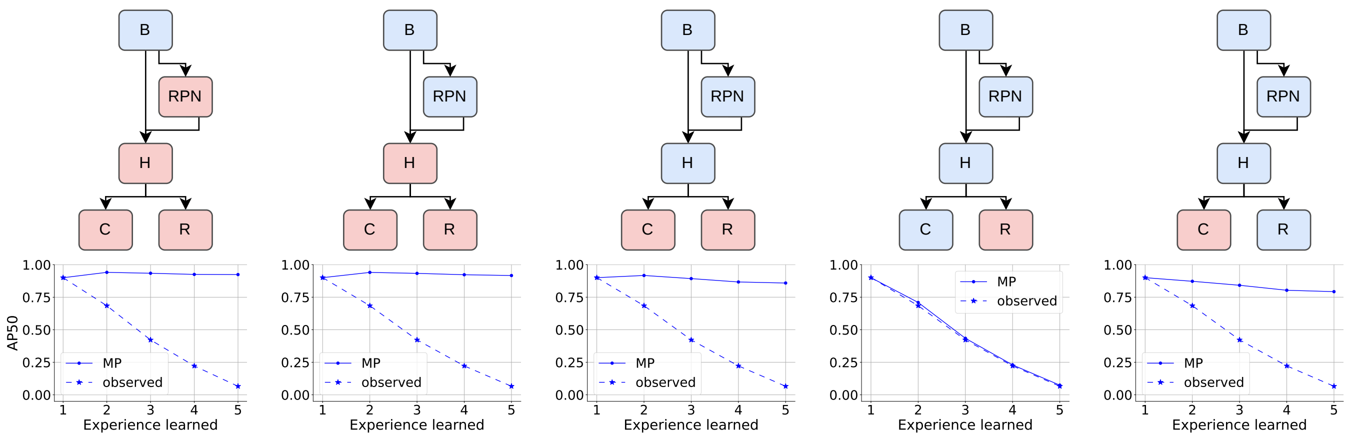
B. Benchmark details

As an agent navigates through an environment, it encounters categories of objects it has previously seen [1]. The agent revisits certain categories more frequently than others, determined by their presence in the environment. This phenomenon of re-occurrence of old categories is called Natural Replay [4]. The disparity in the degree of replay between different categories distorts the model's measures of forgetting. A rare object in the flow is prone to be forgotten, while proficiency in recognizing a commonly encountered object tends to improve over time [4]. Evaluating the genuine forgetting of a model in OOD requires training it in a learning scenario without Natural Replay. For this reason, we performed our experiments on the benchmark EgoObjects Continual Learning (CL) Instance [6], which is free of Natural Replay.

EgoObjects CL Instance [6]: 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.

The dataset is divided into 5 different experiences $(E_1, E_2, \dots E_5)$. Access to images from previous experiences is prohibited, ensuring that only the current experience is used for training. Previous experiences share no common main object instances with later experiences, resulting in a Natural Replay-free benchmark ideal for evaluating forgetting.

Online constraints: The benchmark on EgoObjects is originally introduced for continual object detection, where several



(a) (b) (c) (d) (e)

Figure 3. Performance comparison of *Observed* and *Module Probed (MP)* performances across five 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 (d). In all other configurations, the model successfully readjusts. These observations strongly suggest that the Softmax layer in **C** is the main source of forgetting within the architecture.

epochs are possible on each experience [6]. To respect the *online constraint* specific to the field of study in which we position ourselves, we follow the learning scenario outlined by the benchmark but allow a *single training epoch* on each of the 5 experiences. In addition, following [1], [3], [4], we build a validation set by performing a 1/16 split of the training set. The validation set specific to the first experience E_1 is used to evaluate model forgetting in our Module Probing protocol as D_1. The four latter experiences (E_2, E_3, E_4, E_5) constitute D_2. For a given configuration of frozen modules, we evaluate the model at the end of each experience, this gives four intermediate forgetting measures on D_2.

Training details: We train a Faster R-CNN [2] with a Mobile-Net [10] as Backbone, pre-trained on the COCO dataset [11]. The model is trained by stochastic gradient descent with a learning rate of 0.01, a weight decay of 10^{-5} and a score threshold of 0.001. The model is evaluated using the Average Precision metric with an IoU (Intersection over Union) threshold of 0.5 (AP50).

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 the following modules. This results in five different freezing configurations, illustrated at the top of Figure 3.

C. Results

In the following, we present the results of Module Probing on the EgoObjects CL Instance benchmark [6]. For a given configuration of frozen modules, we show two evaluation measures. The *observed* AP50 is the performance of the model trained in an incremental way on each experience E_i. *Module Probed* (*MP*) performance is evaluated after readjusting non-frozen modules on the reference experience E_1 at the end of training on $E_i, i \geq 2$ as explained in the Module Probing protocol in III-A. The results are shown in Figure 3.

During online learning, the Faster R-CNN loses knowledge of previous data: the *observed* performance curve decreases progressively as the model is trained on other data in E_2, E_3, E_4 and E_5.

In the first three configurations (3(a), 3(b), 3(c)), the *MP* AP50 is readjusted compared to *observed*. This suggests that the frozen modules have retained information relating to the first experience E_1. The forgetting effect in the Faster R-CNN is not localized in the first three modules Backbone, RPN, and Linear Head. Hence, it is unnecessary to readjust these modules to obtain good performance on earlier data.

In the forth freeze configuration where the Box Classifier is additionally frozen (3(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 E_1, the rest of the architecture is frozen. This is the only configuration among five where the Box Classifier is frozen.

In the last configuration (3(e)), the Box Classifier is the only module retrained on E_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 E_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 the field of online classification [12], the Softmax layer has been criticized in numerous works [12]–[14]. This layer induces a classification bias towards more recent classes, leading to poor classification performance, despite limited intermediate representation forgetting [7].

This finding links the manifestation of forgetting in the domains of classification and object detection in online learning. In the following section, we modify the Faster R-CNN at the level of the Box Classifier only to address the observed forgetting of this architecture in an OOD scenario.

IV. Forgetting Compensation in Faster R-CNN for OOD

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.

Based on these results, we modify the Softmax layer in the Faster R-CNN architecture to enhance its information retention. In this section, we detail the new architecture and demonstrate its performance on three OOD benchmarks.

A. Mitigating forgetting with Label Trick

In a class-imbalanced training context, the Softmax classifier is biased towards over-represented classes [15]. In OOD, recent classes tend to be more represented than old classes. Thus, Softmax favors new classes over old ones, perceived as knowledge forgetting [12], [14].

To overcome the problem of Softmax bias in Faster R-CNN, we replace the original cross-entropy loss for classification with the Label Trick (LT) [12]. The loss is the following:

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

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.

Old classes are absent in the current minibatch. By exclusively considering outputs corresponding to classes in the current minibatch, the training avoids excessively penalizing logits for old classes.

Comparison between Softmax and LT: In the following, we compare the performances of the traditional Softmax layer with the LT layer for two main strategies for OOD: *Incremental* and *Experience Replay (ER)* [16]. *Incremental* involves training a Faster R-CNN over the data stream without taking precautions to prevent catastrophic forgetting. *ER* augments the current batch with old images that have been stored in an external memory buffer. ER allows training on new data from the stream as well on old data to preserve knowledge and has shown to be particularly effective in OOD [4]. We assess the advantages of using the LT layer for both strategies: *Incremental* and *ER*.

B. Experimental setting

1) Datasets: In section III, we applied our Module Probing protocol on the Natural Replay-free benchmark **EgoObjects CL Instance** [6]. In this section, we introduce two additional benchmarks with Natural Replay to understand our method's effectiveness in such scenarios.

EgoObjects CL Category [6]: 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.

OAK [1]: This benchmark was introduced in a pioneering work on OOD [1]. It features an egocentric video stream set covering nine months in the life of a student wandering in a campus. The dataset is a subset of the KrishnaCam dataset, consisting of 7.6 million frames divided into 460 video clips, for a total duration of 70.2 hours. The dataset contains 105 object categories. We split the dataset in 20 experiences.

We use to the methodology established by recent works in OOD for the allocation of training and test data [1], [3], [4] 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 I the Natural Replay Score (NRS) [4] 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.

2) Evaluation metrics: We use two evaluation metrics to assess the performance of online object detectors [1]. Let CAP_t denotes the AP at IoU 0.5 at evaluation step t and T the total number of experiences.

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

$$FAP = CAP_T \tag{2}$$

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

$$CAP = \frac{1}{T}\sum_{t=1}^{T} CAP_t \tag{3}$$

3) Implementation details: dIn each experiment, we use the same training setup as described in Section III-B. For the Incremental strategy, the batch size is set to 8. For ER, each batch of 8 images is augmented with 8 images randomly sampled from memory, forming a training batch of 16 images.

C. Results

Table I shows the performance of the *Incremental* and *ER* [16] strategies when using the Softmax or the LT layer. All benchmarks demonstrate that incorporating the LT layer into the Faster R-CNN architecture yields superior performance compared to the traditional Softmax. Notably, the LT layer shows promising results on EgoObjects CL Instance, gaining 28.7 FAP points and 12.5 CAP points for the *Incremental* strategy. On EgoObjects CL Category with a moderate level of Natural Replay ($NRS = 0.51$), LT significantly improves both FAP and CAP compared to using the Softmax layer: FAP increases from 44.1 to 63.8, and the CAP from 39.5 to 48.3. On OAK with high Natural Replay ($NRS = 0.92$), the LT layer yields a FAP of 31.1 and a CAP of 22.7, outperforming Softmax's FAP of 29.1 and CAP of 21.1.

When evaluating the replay method ER, the integration of the LT layer enhances model performance by an average of 1.4 FAP points and 0.8 CAP points across all three benchmarks.

Table I

COMPARISON OF FINAL AVERAGE PRECISION (FAP) AND CONTINUAL AVERAGE PRECISION (CAP) USING SOFTMAX OR LABEL TRICK (LT) LAYERS FOR INCREMENTAL AND ER STRATEGIES. EVALUATED ON EGOOBJECTS CL INSTANCE, EGOOBJECTS CL CATEGORY, AND OAK BENCHMARKS. HIGHER NATURAL REPLAY SCORES (NRS) INDICATE MORE FREQUENT CLASS REVISITS. ↑ INDICATES HIGHER IS BETTER.

		EgoObjects CL Instance ($NRS = 0$)		EgoObjects CL Category ($NRS = 0.51$)		OAK ($NRS = 0.92$)	
		FAP (↑)	CAP (↑)	FAP (↑)	CAP (↑)	FAP (↑)	CAP (↑)
Incremental	Softmax	27.3	25.2	44.1	39.5	29.1	21.1
	LT	**56.0**	**37.7**	**63.8**	**48.3**	**31.1**	**22.7**
ER [16]	Softmax	89.8	54.1	77.1	57.3	34.7	26.9
	LT	**90.8**	**54.6**	**78.8**	**58.2**	**36.3**	**28.0**

These results highlight the effectiveness of modifying the Softmax layer, as discussed in Section III, in effectively addressing forgetting in the Faster R-CNN across both replay-free and replay-based strategies.

D. Discussion

The LT layer demonstrates remarkable effectiveness, particularly on the most challenging benchmarks. On EgoObjects CL Instance ($NRS = 0$), LT doubles the performance of an Incremental Faster R-CNN, highlighting its robustness in Natural Replay-free scenarios.

In the presence of Natural Replay, LT still enhances model performances, albeit to a lesser extent. On OAK ($NRS = 0.91$), the FAP gap between Softmax and LT is around 2 points for Incremental and ER strategies. This observation is expected, as the presence of Natural Replay tends to balance training between old and new classes. In this case, the Softmax classifier is less biased to recent classes as old classes are revisited. ER reproduces this phenomenon as it introduces an artificial replay in the training loop.

The LT layer restricts the modifications to the weights linked to old classes. In the absence of replay, either Natural Replay or ER, the introduction of our loss in the Faster R-CNN efficiently compensates for the forgetting induced by the lack of revisiting of old classes.

V. CONCLUSION

This paper carries out an in-depth study of the catastrophic forgetting perceived in the Faster R-CNN architecture in an Online Object Detection scenario. The proposed Module Probing methodology measures forgetting at different levels of the architecture. Evaluation using Module Probing revealed that the softmax classification layer is the main source of forgetting in the architecture. We therefore proposed the first replay-free baseline for OOD, which consists of modifying the architecture at the classification layer. Our method greatly improves the performance of Faster R-CNN in OOD by limiting its forgetting, especially on more challenging benchmarks that present little revisit of old classes in the data stream.

Acknowledgements 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] 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.

[2] 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.

[3] J. Z. Wu, D. J. Zhang, W. Hsu, M. Zhang, and M. Z. Shou, "Label-efficient online continual object detection in streaming video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19246–19255, 2023.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, "Online continual learning in image classification: An empirical survey," *Neurocomputing*, vol. 469, pp. 28–51, 2022.

[13] 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.

[14] G. Liang, Z. Chen, Z. Chen, S. Ji, and Y. Zhang, "New insights on relieving task-recency bias for online class incremental learning," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[15] 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.

[16] 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.