ONLINE BIAS CORRECTION FOR TASK-FREE CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Task-free continual learning is the machine-learning setting where a model is trained online with data generated by a non-stationary stream. Conventional wisdom suggests that, in this setting, models are trained using an approach called experience replay, where the risk is computed both with respect to current stream observations and to a small subset of past observations. In this work, we show both theoretically and empirically how experience replay biases the outputs of the model towards recent stream observations. Moreover, we propose a simple approach to correct for this bias online, by changing the way the output layer of the model is optimized. We show that our approach improves significantly the learning performance of experience-replay approaches over a number of different datasets. Our findings suggest that, in contrast to stationary machine-learning problems, the output layer of a model should be optimized separately from its preceding layers when performing experience replay.

1 INTRODUCTION

In broad terms, *continual learning* is the process of incrementally aggregating knowledge from data that are generated by a non-stationary distribution (Lee et al., 2019; Riemer et al., 2019). The main motivation for studying continual learning is to give artificial learners the ability to learn as biological learners do—perpetually updating and refining their body of knowledge under changing environmental conditions (Silver et al., 2013). The inability of artificial learners to learn continually is caused by the fact that, in practice, they overwrite previously learned knowledge whenever they encounter new information. This phenomenon is called *catastrophic forgetting* (McCloskey & Cohen, 1989; French, 1999).

In this paper, we focus specifically on *task-free* continual learning (Aljundi et al., 2019c). In this setting, the data are presented to the learner in small minibatches, and this setting is agnostic to the way the data distribution changes over time. In other words, we assume no knowledge about whether the distribution is piecewise-stationary (such as when there are distinct tasks being learned), or whether the distribution changes continuously over time (Aljundi et al., 2019c). Most task-free continual learning approaches make use of a memory which can store a small percentage (typically 10% or less) of all observed data instances. The data instances stored in memory are replayed in future steps to mitigate catastrophic forgetting. This simple paradigm, called *replay-based* continual learning, is surprisingly effective in task-free settings. Furthermore, it is also supported by findings from the field of neuroscience, in relation to how biological learning takes place (Marr, 1971; Ji & Wilson, 2007; Liu et al., 2019).

A number of continual learning methods tend to make predictions that are biased towards recently observed data (Buzzega et al., 2021; Mai et al., 2021). Several strategies have been proposed to deal with this prediction bias (also called recency bias). Unfortunately, most of them are not applicable to task-free continual learning, since they have been designed for continual learning settings that consist of a task sequence, and they require knowledge of which classes the current task consists (Wu et al., 2019; Belouadah & Popescu, 2019; Buzzega et al., 2021). One approach which is applicable in task-free continual learning is proposed in Mai et al. (2021), but it can only be performed after the end of training, hence the learner's predictions during the entirety of training remain biased.

In this paper, we propose a simple approach that performs online bias correction for task-free continual learning. Our contributions are as follows: a) We formally illustrate that the conventional

paradigm of model training in task-free continual learning overweights the importance of current stream observations (Section 3.2), and we speculate that this overweighting is a cause of prediction bias of continual learners; b) We propose of a novel metric to quantify prediction bias, and we show that this bias can be effectively mitigated by appropriately modifying only the parameters of the final layer of the model, after the end of training (Section 3.4); c) We propose a novel approach called Online Bias Correction (OBC), which maintains an unbiased model online, throughout the entire duration of learning (see Figure 1 for an illustration); d) We evaluate the performance of OBC extensively, and we show that it significantly improves a number of other task-free continual learning methods, over multiple datasets.

2 BACKGROUND

2.1 TASK-FREE CONTINUAL LEARNING

We define *task-free* continual learning as the online optimization of a model via small minibatches that are sampled from a nonstationary stream. In contrast to other continual learning settings, no strong assumptions are made about the nature of the distributional nonstationarity in task-free continual learning (Aljundi et al., 2019c). Other continual learning settings, such as *task-incremental* and *class-incremental* continual learning assume a data distribution that is piecewise stationary, hence only changes at discrete points in time (Van de Ven & Tolias, 2019).

Previous work in task-free continual learning is mostly focused on replay-based methods (Aljundi et al., 2019d; Jin et al., 2021). The dominant replay paradigm is called experience replay (ER) (Isele & Cosgun, 2018; Chaudhry et al., 2019). According to the ER paradigm, each minibatch of observations received by the learner is combined with another minibatch of equal size sampled from the memory. The model is then trained for one step with the combined stream-and-memory minibatch. Moreover, the memory is typically maintained by an online memory-population algorithm called reservoir sampling (Vitter, 1985b).

There are multiple variants of the ER method. For instance, one approach called Maximally-Interfered Retrieval (MIR) replays instances that are going to be interfered the most by the current minibatch of new observations. Another approach called Class-Balancing Reservoir Sampling (CBRS) (Chrysakis & Moens, 2020b) modifies the memory population algorithm to ensure that the memory remains balanced. There also other approaches that deviate from the ER paradigm, such as Greedy Sampler and Dumb Learner (GDUMB) (Prabhu et al., 2020b), which only trains the model using data stored in memory, or Asymmetric Corss-Entropy (ACE) (Caccia et al., 2021), which uses a modified version of the cross-entropy loss to prevent the drift of latent representations.

2.2 COMPUTATIONAL COST

An important matter in task-free continual learning is computational cost. Since practical applications might have to deal with large amounts of data, the design of task-free continual learners should ensure they are tractable. In practical terms, let us assume that a model has to learn from a stream of size n. Moreover, we assume that the larger the stream the larger size m of the memory will be. In real-world applications, the difference between an $O(n)$ learning algorithm, and an $O(mn)$ algorithm might be enormous. Hence, in this work we will only focus on learning algorithms whose computational cost per incoming batch is independent from the memory size m, so that the computational complexity of learning from the entire stream is $O(n)$.

2.3 BIAS CORRECTION IN TASK-FREE CONTINUAL LEARNING

To the best of our knowledge, there only one approach explicitly designed to correct for prediction biases. Mai et al. (2021) propose learning a model using conventional experience replay, and after the entire stream has been observed, they replace the final linear layer of the model with a nearest-class-mean (NCM) classifier computed using all data stored in memory. Moreover, they demonstrate that this approach is effective in increasing the final accuracy of the model. However, in many real-world applications, there is a need for a model that learns and performs inference at the same time, and that model should remain unbiased all the time. In such a setting, the NCM approach would

have to be applied after every update of the model, and since it needs to make a full pass over the memory, it would be computationally very expensive.

3 METHODOLOGY

3.1 NOMENCLATURE

One major difference of this work compared to previous ones, is that we view the neural network as two separate components, instead of one. Specifically, we will call the output layer of a neural network the *classifier*, and we will denote it as the parameterized function $c(\boldsymbol{x};\boldsymbol{\theta}_c)$. Moreover, we will refer to the set of layers preceding the classifier as the *feature extractor*, and we will similarly denote it as $g(\boldsymbol{x};\boldsymbol{\theta}_g)$. Using this notation, we can represent the full neural network h as the composition of g and c, that is

$$h(\boldsymbol{x};\boldsymbol{\theta}_h) \triangleq c\big(g(\boldsymbol{x};\boldsymbol{\theta}_g);\boldsymbol{\theta}_c\big), \quad \text{where } \boldsymbol{\theta}_h \triangleq \{\boldsymbol{\theta}_g, \boldsymbol{\theta}_c\}. \tag{1}$$

An important distinction between the classifier and the feature extractor, is that classifier has low learning capacity because it consists of only one layer, while the feature extractor has high learning capacity because it is composed by multiple layers. Hence, given enough data, the feature extractor can learn more complex representations, but, in low-data scenarios, it is also more prone to overfitting, compared to the classifier.

3.2 DATA-SAMPLING BIAS

At a high level, the optimization process during task-free continual learning is very simple. At each step t, the learner receives a small minibatch of b observations $\mathbf{S}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{b}$ from the stream. The learner then samples another minibatch of equal size $\mathbf{R}$ from its memory $\mathbf{M} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$, and performs an update step over the model parameters using gradients calculated with respect to $\mathbf{S}_t \cup \mathbf{R}$. Finally, the learner updates its memory with respect to $\mathbf{S}_t$, using reservoir sampling (Vitter, 1985a), or some other memory population algorithm. We argue that this optimization process has a data sampling bias, which is caused by the fact that the stream minibatch $\mathbf{S}_t$ and the replay minibatch $\mathbf{R}$ contribute equally in the optimization of the model.

An intuitive way to understand this data sampling bias is to consider a learner with an infinite memory. Let us assume, that the learner has encountered a data instances in the past—all of which have been stored in the memory for replay—and that $b \ll a$, where b is the size of the minibatch of observations, $\mathbf{S}_t$.[1] Assuming that all observations, current and past, are equally important, we will say that there is no data sampling bias if all of them contribute equally to the optimization of the model. Put another way, the probability of using each instance in the model-optimization step should be the same, regardless of whether that instance is in the current stream minibatch $\mathbf{S}_t$, or stored in memory. In practical terms, the model should be updated with a minibatch $\mathbf{B}$ sampled uniformly at random from the set of all observed instances (in effect, this set is the concatenation of the memory and the current stream minibatch $\mathbf{M} \cup \mathbf{S}_t$). Therefore, on average, the fraction of instances in the minibatch $\mathbf{B}$ that come from $\mathbf{S}_t$ is $b/(a+b)$, and the remainder, $a/(a+b)$, is the fraction of instances in $\mathbf{B}$ that come from the memory $\mathbf{M}$. At a high level, this weighting means that the more extensive the past of the learner is, the more important a role it should play in the updates of the model.

In experience replay, however, the past and the present are weighted equally. This equal weighting is essentially a data-sampling bias that favors the present over the past. In turn, this data sampling bias causes the model to produce outputs that are biased towards the stream data it recently observed. A hypothesis arising from this discussion is that prediction biases could be neutralized if the learner had a memory of infinite size and if it sampled the minibatches used to update the model by appropriately weighting past and present.

Unfortunately, in practical applications, it is not possible to have a memory of infinite size. Moreover, as the learner observes more and more data it is $b \ll a$, which means that $a/(a+b) \simeq 1$ and $b/(a+b) \simeq 0$. Therefore, on average, the minibatches $\mathbf{B}$ used to train the model will contain almost exclusively data from the memory and almost no data from the current stream batch $\mathbf{S}_t$. Since in

[1] This is a reasonable assumption to make for task-free continual learning, since, for the learner, the amount of data encountered in the past vastly outnumbers the amount of current new observations.

Table 1: We compare the final accuracy (Acc.) and the bias of ER, MRO, and ER with post-training bias correction (ER+BC), over four datasets. All entries are 95%-confidence intervals over 15 runs.

	FashionMNIST		CIFAR-10		CIFAR-100		tinyImageNet	
	Acc.	Bias	Acc.	Bias	Acc.	Bias	Acc.	Bias
ER	83.4 ± 0.5	0.9 ± 0.2	45.5 ± 1.7	17.0 ± 2.4	19.3 ± 0.7	33.8 ± 2.0	13.5 ± 0.5	25.3 ± 1.7
MRO	83.7 ± 0.5	0.4 ± 0.3	38.9 ± 1.6	8.7 ± 2.7	14.7 ± 0.6	17.0 ± 2.2	10.7 ± 0.4	12.6 ± 1.3
ER+BC	84.8 ± 0.2	0.4 ± 0.1	54.4 ± 0.8	6.6 ± 0.5	27.4 ± 0.4	6.3 ± 0.4	20.4 ± 0.3	5.1 ± 0.3

practical applications the size of the memory is much smaller to the size of the stream, this weighting scheme will arguably lead to overfitting the memory data.

In the following section, we propose a metric to quantify the prediction bias of a model. In addition, we show that this bias is trivial to correct after the continual learning has concluded.

3.3 Quantifying Prediction Bias

We propose to quantify prediction bias with respect to a set of unseen data. Let $\mathbf{T} = \{(\boldsymbol{x}_i, y_i)\}_i$ be such a set, and by $\boldsymbol{y}_i$, we will denote the one-hot representation of y_i. We define the expected vector of prior class probabilities with respect to this test set as $\boldsymbol{p} = \mathbb{E}[\boldsymbol{y}_i]$. Moreover, we define the the expected vector of model predictions as $\boldsymbol{q} = \mathbb{E}\big[h(\boldsymbol{x}_i; \boldsymbol{\theta}_h)\big]$. Note that, both expectations are computed with respect to the instances of the unseen test set.

We can quantify the prediction bias of the model by measuring the discrepancy between $\boldsymbol{p}$ and $\boldsymbol{q}$. Since both $\boldsymbol{p}$ and $\boldsymbol{q}$ are vectors of probabilities, we propose the use of the Jensen-Shannon divergence (Lin, 1991), which is a symmetric divergence measure between two probability distributions. It is defined as

$$\text{JS}(\boldsymbol{p} \,||\, \boldsymbol{q}) = \frac{1}{2}\left[\sum_{j=1}^{c} p_j \log \frac{\mu_j}{p_j} + \sum_{j=1}^{c} q_j \log \frac{\mu_j}{q_j}\right] \tag{2}$$

where $\boldsymbol{\mu} = 1/2(\boldsymbol{p} + \boldsymbol{q})$ is the average between $\boldsymbol{p}$ and $\boldsymbol{q}$, and j is the index over the classes $1, \ldots, c$, that are present in the test set. When the model predictions are not biased, $\boldsymbol{p}$ and $\boldsymbol{q}$ will be very similar, and the divergence will be close to zero. In the opposite case, the more dissimilar $\boldsymbol{p}$ and $\boldsymbol{q}$ are, the more the divergence will increase.

3.4 Post-Training Bias Correction

At this point, we will show that we can significantly reduce the prediction bias of a task-free continual learner if we appropriately change the parameter vector $\boldsymbol{\theta}_c$ of the classifier after the entire stream has been observed. We will compare three approaches over four datasets (we follow the experimental settings described in Section 4.1). We report both the accuracy and the bias (according to Eq. 2) after the end of the stream. We compare regular experience replay (ER), which weights equally its present and past, and memory-replay only (MRO), which uses only data stored in memory to train the model. Moreover, we also evaluate a version of ER with post-training bias correction (ER-BC). To correct the bias of the model, we use the data stored in memory to train only the classifier of the model until convergence. The feature extractor of the model remains unchanged.

The results of this comparison are presented in Table 1. We observe that MRO is less biased than ER in all four datasets. On the other hand, despite being more biased, ER outperforms MRO in CIFAR-10, CIFAR-100, and tinyImageNet. As we hypohtesized at the end of Section 3.2, training a model using only the data stored in memory will reduce the prediction bias of that model, but will also lead to overfitting. On the contrary, ER+BC is less biased than ER, and achieves significantly higher accuracy values on all four datasets.

These results are evidence that prediction bias can be corrected just by changing the way the classifier is trained. Moreover, because the classifier consists of just one layer, it has lower learning capacity, and, therefore, overfits much less than if we were to train the entire model using only the data in memory (as is the case with MRO). In the following section, we exploit the findings of this section to propose an algorithm that maintains an unbiased model throughout the entire stream.

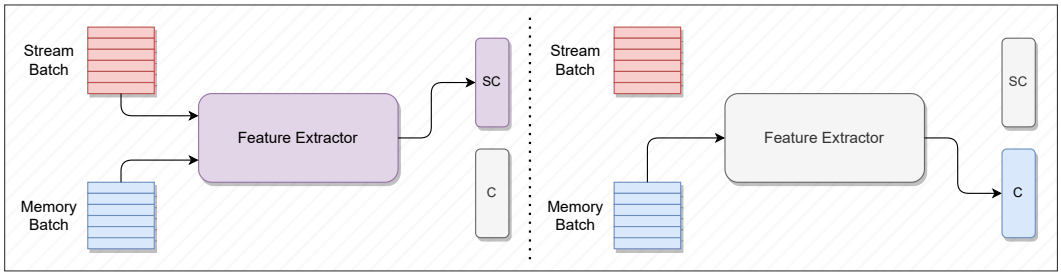

Figure 1: An illustration of Online Bias Correction (OBC). At first (left), the feature is extractor is trained using both a stream and a replay minibatch (as in conventional experience replay), with the help of a surrogate classifier (SC). Afterwards (right), the classifier (C) is trained using only a minibatch sampled from the memory, while the feature extractor remains unchanged. This two-step process is repeated for every incoming stream minibatch. The color of a model component illustrates with respect to which data its gradients are calculated. Lack of color for a component means that it is not trainable.

3.5 Online Bias Correction

Our approach is called *Online Bias Correction* (OBC) and acts as a wrapper around other task-free continual learning methods that perform experience replay. At first, a generic task-free continual learning method performs its training step. Such a training step, typically includes receiving a minibatch of observations $\mathbf{S}_t$ from the stream (Line 1), sampling another minibatch $\mathbf{R}$ from memory (Line 2), and performing a training step using the combination of the two minibatches (Line 3). Note that only the parameter vector of the feature extractor (FE), and that of the surrogate classifier (SC) are updated during this training step. Next, the learner decides which of the new observations in $\mathbf{S}_t$ to store in memory, and which ones to discard (Line 4).

The following four steps (i.e., Lines 5-8) are the ones that OBC introduces. First, the feature extractor is frozen (Line 5)—that is, in the update step that follows, there will be no gradients with respect to its parameter vector calculated in the backward pass, and thus its parameter vector will not be updated. Next, a new minibatch $\mathbf{R}^*$ is sampled from the memory (Line 6). In the following step, this memory minibatch is feedforwarded through the feature extractor and then the classifier (C), and only the parameter vector of the classifier, is updated with the resulting gradients (Line 7). Since the classifier is trained using only memory data, in order to reduce overfitting, we make use of label smoothing (Szegedy et al., 2016) to act as regularization. Finally, the feature extractor is unfrozen, meaning that its parameter vector is trainable again (Line 8).

Algorithm 1 Online Bias Correction

Stream minibatches $\mathbf{S}_t$, $t = 1, 2, \ldots,$
Memory $\mathbf{M}$, Feature Extractor (FE),
Classifier (C), Surrogate Classifier (SC)

1: **for** each stream minibatch $\mathbf{S}_t$ **do**
2: Sample a memory minibatch $\mathbf{R} \sim \mathbf{M}$
3: Train the FE and the SC using $\mathbf{R} \cup \mathbf{S}_t$
4: Perform memory population using $\mathbf{S}_t$
5: Freeze the FE
6: Sample a memory minibatch $\mathbf{R}^* \sim \mathbf{M}$
7: Train the C using $\mathbf{R}^*$
8: Unfreeze the FE
9: **end for**

The OBC paradigm was constructed based on the results of the previous section. In particular, we exploit that, as we saw earlier, it is possible to correct the bias by just changing the weights of the classifier. Moreover, the classifier, in the context discussed here, consists of a single layer, hence it is more resistant to overfitting compared to the feature extractor, which is typically a deep neural network of high learning capacity. Therefore, by using the memory data to train the classifier, we reduce the data-sampling bias, and hence, the prediction bias. To further prevent the classifier from overfitting the memory data we propose using label smoothing (Szegedy et al., 2016), which is a technique that adds noise to the labels in order to discourage overconfident predictions. Finally, we note that the design choice to introduce a surrogate classifier is motivated experimentally in Section 4.4.

Table 2: Comparison of various task-free continual learning methods with and without OBC, over four datasets. We report the accuracy after training (Acc.) and the information retention averaged over the stream (Av. IR). All entries are 95%-confidence intervals over 15 runs.

	FashionMNIST		CIFAR-10		CIFAR-100		tinyImageNet	
	Acc.	Av. IR	Acc.	Av. IR	Acc.	Av. IR	Acc.	Av. IR
ER	83.4 ± 0.5	86.5 ± 0.2	45.5 ± 1.7	60.0 ± 0.4	19.3 ± 0.7	27.0 ± 0.3	13.5 ± 0.5	15.4 ± 0.2
+OBC	84.9 ± 0.3	89.8 ± 0.1	54.3 ± 1.1	67.5 ± 0.4	25.1 ± 0.6	35.2 ± 0.4	16.9 ± 0.5	21.7 ± 0.2
MIR	83.7 ± 0.4	87.5 ± 0.2	47.1 ± 1.5	59.1 ± 0.7	18.8 ± 0.7	27.4 ± 0.4	12.4 ± 0.7	15.8 ± 0.2
+OBC	84.9 ± 0.3	89.7 ± 0.1	53.4 ± 1.3	66.8 ± 0.4	23.8 ± 0.6	34.9 ± 0.4	15.5 ± 0.6	21.7 ± 0.2
CBRS	82.9 ± 0.5	84.8 ± 0.2	44.9 ± 1.8	57.6 ± 0.4	19.2 ± 0.7	25.7 ± 0.3	13.4 ± 0.6	14.8 ± 0.2
+OBC	84.5 ± 0.3	89.2 ± 0.1	53.4 ± 1.1	66.7 ± 0.4	25.4 ± 0.6	35.0 ± 0.3	17.0 ± 0.5	21.6 ± 0.2
GMED	83.9 ± 0.7	87.0 ± 0.2	46.2 ± 1.8	60.4 ± 0.5	19.8 ± 0.8	27.2 ± 0.4	13.7 ± 0.7	16.1 ± 0.3
+OBC	85.1 ± 0.4	90.1 ± 0.3	54.7 ± 1.3	67.9 ± 0.5	25.4 ± 0.8	35.4 ± 0.4	17.1 ± 0.6	21.9 ± 0.3

4 Experiments

4.1 Experimental Settings

Datasets We use four datasets of varying difficulty. The FashionMNIST dataset (Xiao et al., 2017) contains 60,000 grayscale images of clothing items split in 10 classes. CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) each contain 50,000 color images, with the ones in CIFAR-10 being divided in 10 classes, and the ones in CIFAR-100 in 100 classes. Finally, tinyImageNet (Le & Yang, 2015) contains 100,000 color images of 200 different classes. The tinyImageNet dataset is the most challenging dataset widely used in task-free continual learning literature, mainly due to its large number of classes (200) and the small number of data instances per class (500). We follow the majority of past work (Aljundi et al., 2019e;b; Chrysakis & Moens, 2020a) by not using data augmentation in our experiments.

Methods Experience replay (ER) (Isele & Cosgun, 2018; Chaudhry et al., 2019) is the most fundamental continual learning baseline. It performs replay using a memory that is populated using reservoir sampling (Vitter, 1985a). Memory-replay only (MRO) also uses reservoir sampling to populate the memory, but instead of using both stream and memory data, it only trains the model using data from the memory. Maximally-interfered retrieval (MIR) (Aljundi et al., 2019b) is an extension of ER that replays the instances which would experience the largest loss increases, if the model were to be updated using only the current minibatch of observations. Class-balancing reservoir sampling (CBRS) (Chrysakis & Moens, 2020a) uses a memory population algorithm that attempts to maintain the memory as class-balanced as possible at all times. Greedy sampler and dumb learner (GDUMB) (Prabhu et al., 2020a) also uses a class-balancing memory population algorithm and trains the model using only data stored in memory.[2] Gradient-based memory editing (GMED) (Jin et al., 2021) edits the data stored in memory in order to increase their loss values and make them more challenging. Finally, asymmetric cross entropy (ACE) (Caccia et al., 2021) modifies the traditional cross-entropy loss with class-masking that reduces representation drift during continual learning.

Architectures and Hyperparameters Similar to previous work (Lopez-Paz & Ranzato, 2017; Aljundi et al., 2019a), we use a reduced ResNet-18 architecture (He et al., 2016) for CIFAR-10, CIFAR-100, and tinyImageNet. For the simpler FashionMNIST, we use a simple convolutional neural network (CNN). For more information on these architectures, please refer to the appendix. Following previous work (Aljundi et al., 2019b; Jin et al., 2021), we use a learning rate of 0.1 when using the reduced ResNet-18 architecture. When using the simpler CNN, we use a learning rate of 0.03. The stream and replay batch sizes were both set to 10, in accordance with past work. The batch size of OBC was set to 50 (please refer to Section 4.5 for a sensitivity analysis.) Past work typically uses memory sizes in the range of 1%–10% of the size of the stream. Hence, unless explicitly mentioned otherwise, we set the memory size to 1000 for both FashionMNIST (approximately 2%)

[2] We adapt GDUMB for use in task-free continual learning (in this setting, the learner should always be available for inference, while in (Prabhu et al., 2020a), the model is trained only after the entire stream is observed).

Table 3: Comparison of various bias-correction methods for three memory sizes on CIFAR-10. We report the accuracy after training (Acc.), the information retention averaged over the stream (Av. IR), and the bias of each approach. All entries are 95%-confidence intervals over 15 runs.

	500			1000			2500		
	Acc.	Av. IR	Bias	Acc.	Av. IR	Bias	Acc.	Av. IR	Bias
ER	37.4 ± 1.4	55.0 ± 0.3	28.9 ± 2.4	45.5 ± 1.7	60.0 ± 0.4	18.1 ± 2.8	56.1 ± 1.8	64.3 ± 0.5	8.3 ± 1.7
GDUMB	37.5 ± 1.2	54.5 ± 0.3	16.8 ± 2.2	42.2 ± 1.7	57.9 ± 0.3	10.8 ± 2.7	49.3 ± 1.5	61.3 ± 0.4	6.6 ± 1.6
MRO	34.4 ± 1.3	54.9 ± 0.3	9.2 ± 3.3	38.2 ± 1.6	58.1 ± 0.3	10.5 ± 2.7	46.0 ± 1.5	61.6 ± 0.3	7.7 ± 1.7
ACE	46.7 ± 1.4	63.3 ± 0.4	4.6 ± 1.6	51.6 ± 1.9	65.3 ± 0.5	5.3 ± 2.2	53.8 ± 1.6	67.5 ± 0.4	5.0 ± 1.4
ER+OBC	46.8 ± 1.3	63.4 ± 0.4	1.3 ± 0.3	54.3 ± 1.0	67.6 ± 0.4	0.9 ± 0.2	61.5 ± 1.3	71.0 ± 0.4	1.0 ± 0.4

and CIFAR-10 (2%), and to 2500 for CIFAR-100 (5%) and 5000 for tinyImageNet (5%).[3] For OBC, and only when training the classifier, we use a batch size of 50, and a label-smoothing[4] factor of 0.5 (we perform a sensitivity analyis of these two hyperparameters in Section 4.5). Method-specific hyperparameters were set to the values given in their respective papers.

Evaluation Metrics We use the accuracy computed on the unseen test set after the entire stream has been observed. In order to have a more complete picture than what the post-training accuracy would give us, we also propose to use the information retention metric (accuracy computed with respect to past observations) proposed in Cai et al. (2021), averaged over the entire stream. This metric is a form of continual evaluation that evaluates how well each method performs, not only after the end of learning, but during its entire duration. Continual evaluation is crucial in real-world applications where a model needs to train and perform inference interchangeably. We also use the bias metric we proposed earlier (please refer to Section 3.4).

4.2 Applying OBC

At first we apply OBC to four state-of-the-art, task-free continual learning methods. Namely, we compare the performance (final accuracy and information retention averaged over the stream) of ER, MIR, CBRS, and GMED, with and without OBC on four different datasets (see Table 2). We observe that OBC improves both the learning performance of each method over the continuum (Av. IR), and their final accuracy, for all four datasets. The performance boost that OBC provides is especially evident in CIFAR-10, CIFAR-100, and tinyImageNet. We note that methods that perform post-training bias correction (as the one we proposed in Section 3.4, or the NCM approach proposed by Mai et al. (2021)) would only affect the final accuracy of each method, but not the average information retention over the stream, because, by definition, the bias correction takes place after the end of training.

Keeping in mind that OBC only changes the way the classifier is trained, we argue that when applying bias correction to various methods, we can essentially compare the quality of the representations that their feature extractor learns. In the results presented here, there is no clear best method in that respect, but future work could specifically focus on how to make sure that the feature extractor learns better representations.

4.3 Comparison with Other Bias Correction Approaches

At this point, we will compare OBC to three other methods that perform implicit bias correction (in the sense that they were not designed specifically to correct for prediction bias, but they do so anyway). In particular, we compare OBC to GDUMB, MRO (they are not biased by the stream in the first place since they both train models using memory data only), and ACE (it uses both stream and replay data, but with a masked cross-entropy, which is aimed to reduce representation drift, but also does not lead to biased predictions). We also include ER in this comparison, as a biased

[3] To avoid potential confusion, we would like to point out that some previous works (e.g., Aljundi et al. (2019b)) do not report the size of the entire memory, but the memory size divided by the number of classes instead. Along similar lines, the memory sizes used here are 100, 100, 25, and 50, respectively.

[4] We found that when label smoothing is applied in combination with other methods (ER, GDUMB, ACE, etc.) it invariably leads in reduced performance, possibly because it is not applied specifically to the training of the classifier, but to the entire model.

Table 4: Comparison of various bias-correction methods for three memory sizes on CIFAR-100. We report the accuracy after training (Acc.), the information retention averaged over the stream (Av. IR), and the bias of each approach. All entries are 95%-confidence intervals over 15 runs.

	1000			2500			5000		
	Acc.	Av. IR	Bias	Acc.	Av. IR	Bias	Acc.	Av. IR	Bias
ER	12.8 ± 0.6	23.4 ± 0.3	57.2 ± 2.4	19.3 ± 0.7	26.9 ± 0.3	34.7 ± 2.0	23.1 ± 0.8	27.8 ± 0.4	25.5 ± 1.6
GDUMB	11.2 ± 0.5	19.2 ± 0.2	26.2 ± 2.6	15.3 ± 0.7	21.5 ± 0.3	16.4 ± 2.3	18.6 ± 0.6	22.4 ± 0.3	10.2 ± 1.2
MRO	10.2 ± 0.5	19.0 ± 0.2	23.8 ± 3.0	14.7 ± 0.6	21.4 ± 0.3	17.0 ± 2.2	18.1 ± 0.6	22.4 ± 0.3	11.1 ± 1.3
ACE	17.8 ± 0.5	27.0 ± 0.3	8.7 ± 1.2	22.4 ± 0.6	29.3 ± 0.4	6.6 ± 1.1	25.7 ± 0.7	30.2 ± 0.4	4.8 ± 0.5
ER+OBC	19.0 ± 0.6	31.0 ± 0.3	2.5 ± 0.5	25.1 ± 0.5	35.2 ± 0.4	2.7 ± 0.4	30.7 ± 0.5	37.1 ± 0.5	1.8 ± 0.3

Table 5: Comparison of OBC with and without (OBC – SC) a surrogate classifier over two datasets. We report the accuracy after training (Acc.), the information retention averaged over the stream (Av. IR), and the bias of each approach. All entries are 95%-confidence intervals over 15 runs.

	CIFAR-10			CIFAR-100		
	Acc.	Av. IR	Bias	Acc.	Av. IR	Bias
ER	45.5 ± 1.7	60.0 ± 0.4	18.1 ± 2.8	19.1 ± 0.7	27.0 ± 0.3	34.7 ± 2.0
+OBC	54.3 ± 1.0	67.6 ± 0.3	0.9 ± 0.2	25.1 ± 0.5	35.2 ± 0.3	2.7 ± 0.4
+OBC – SC	42.8 ± 2.3	58.3 ± 0.4	6.1 ± 1.3	21.0 ± 0.6	25.6 ± 0.5	6.1 ± 0.5

baseline. The comparison takes place in CIFAR-10 (see Table 3) and CIFAR-100 (see Table 4). We use memory sizes that correspond to 1%, 2%, 5% of the size of CIFAR-10, and 2%, 5%, 10% of the size of CIFAR-100. For CIFAR-10, we can see that OBC outperforms GDUMB and MRO, and is competitive with ACE for very small memories of 500 instances. For the other two memory sizes, OBC outperforms GDUMB, MRO, and ACE, with respect to all three metrics used. In CIFAR-100, OBC outperforms GDUMB, MRO, and ACE, for all three memory sizes. Interestingly, we want to point out that for both CIFAR-10 and CIFAR-100, the gaps in accuracy and average information retention between OBC and the other three bias-correction methods increase with larger memory sizes.

4.4 The Need for a Surrogate Classifier

Finally, we want to explain why we need to introduce a surrogate classifier in the OBC paradigm. We compare the performance of OBC when applied to ER, with and without the use of a surrogate classifier. In the latter case, the computational graph in the left part of Figure 1 flows from the feature extractor to the classifier, but the classifier is not updated with the resulting gradients. The right part of Figure 1 remains unchanged.

The results are presented in Table 5. We observe that both for CIFAR-10 and CIFAR-100, the alternative formulation of OBC without a surrogate classifier performs worse than OBC with a surrogate classifier. These results validate the need for using a surrogate classifier. Since OBC is applied in both cases only for the optimization of the classifier, we can only conclude that not using a surrogate classifier somehow harms the feature extractor. In other words, it appears that the feature extractor learns better representations when it is trained in combination with the biased surrogate classifier than when it is trained combined with the unbiased classifier. Future work focusing on better representation learning in task-free settings could possibly interpret this finding further.

4.5 Hyperparameter Sensitivity

We now examine how sensitive OBC is to its hyperparameters. (Note that other hyperparameters such as the stream batch size are applicable, not directly to OBC, but only to the method that OBC wraps.) In Figure 2, we present the final accuracy of ER+OBC for various values of the OBC batch size (10, 20, 50, 100, 200), and the label-smoothing factor (0.2,. . . ,0.8).[5] The accuracies correspond

[5]We remind the reader, that for all previous experiments we use an OBC batch size of 50 and a label-smoothing factor of 0.5.

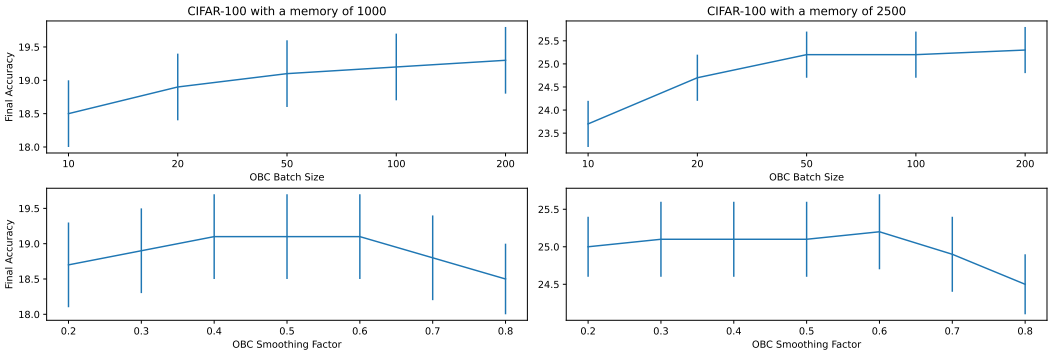

Figure 2: We examine the sensitivity of OBC with respect to its two hyperparameters, namely, the label-smoothing factor, and the batch size. The comparison is performed on CIFAR-100 for two memory sizes. The results are presented as 95%-confidence intervals over 15 runs.

to CIFAR-100 for memory sizes of 1000 and 2500, and are presented as 95%-confidence intervals. The batch-size curves are very similar for both memory sizes. Higher batch sizes are correlated with higher accuracies, but with diminishing returns. For the label-smoothing factor, we can see that relatively large values (0.7 or 0.8) lead to lower accuracies for both memory sizes. In addition, only for the smaller memory of 1000 instances, we also observe that a relatively small label-smoothing factor of 0.2 or 0.3 also leads to slightly reduced accuracies.

4.6 Discussion

As we saw in Section 4.2, OBC performs reliably for multiple methods and over multiple datasets. In Section 4.3, we showed that it also outperforms other methods that implicitly perform bias correction, for different memory sizes, and in two different datasets.

In our view, one important finding of this work is that, in the context of task-free continual learning, we should not necessarily be viewing a neural network as one entity. Instead, we should make a distinction between the feature extractor and the classifier, because of their different learning capacity.

One potential limitation of OBC is that it does not improve the feature extractor of the model. Consequently, the final performance after applying OBC will depend on the quality of the representations learned by the method being wrapped. Therefore, an interesting avenue for future work, is to improve these representations. By using OBC, it is now possible to correct for differences in the classifier, and evaluate in isolation which task-free continual learning methods learn better representations. Another interesting direction for future work would be to investigate the extent to which the data-sampling bias affects the feature extractor of the learner.

5 Conclusion

To summarize, we studied the issue of prediction bias in task-free continual learning. Specifically, we first explained intuitively that this bias is essentially caused by overweighting current stream observations over the past data stored in memory. We then proposed a metric to quantify prediction bias, and showed that our intuitive explanation holds. Moreover, we proposed a simple approach called Online Bias Correction (OBC) that can correct this bias online throughout the entire duration of learning, hence being especially useful for real-world applications in which a model needs to learn and perform inference at the same time. OBC is trivial to implement and can be applied as a wrapper over other task-free continual learning methods in order to increase their accuracy and their information retention, and reduce their prediction bias.

REFERENCES

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In *Advances in Neural Information Processing Systems*, pp. 11849–11860. Curran Associates, Inc., 2019a.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximally interfered retrieval. *Advances in Neural Information Processing Systems*, 32, 2019b.

Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263, 2019c.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, volume 32, pp. 11816–11825. Curran Associates, Inc., 2019d.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems*, 32, 2019e.

Eden Belouadah and Adrian Popescu. Il2m: Class incremental learning with dual memory. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 583–592, 2019.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2180–2187. IEEE, 2021.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2021.

Zhipeng Cai, Ozan Sener, and Vladlen 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*, pp. 8281–8290, 2021.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. Continual learning with tiny episodic memories. In *Workshop on Multi-Task and Lifelong Reinforcement Learning*, 2019.

Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning*, pp. 1952–1961. PMLR, 2020a.

Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning*, pp. 1952–1961. PMLR, 2020b.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

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

David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, no. 1, 2018.

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

Xisen Jin, Arka Sadhu, Junyi Du, and Xiang Ren. Gradient-based editing of memory examples for online task-free continual learning. *Advances in Neural Information Processing Systems*, 34: 29193–29205, 2021.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations*, 2019.

Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

Yunzhe Liu, Raymond J Dolan, Zeb Kurth-Nelson, and Timothy EJ Behrens. Human replay spontaneously reorganizes experience. *Cell*, 178(3):640–652, 2019.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6470–6479, 2017.

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

D Marr. Simple memory: a theory for archicortex. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 262(841):23–81, 1971.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

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

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

Matthew Riemer, Tim Klinger, Djallel Bouneffouf, and Michele Franceschini. Scalable recollections for continual lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, no. 01, pp. 1352–1359, 2019.

Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *2013 AAAI spring symposium series*, 2013.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985a.

Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985b.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 374–382, 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Table 6: (left) A simple convolutional block; (middle) The Convolutional Neural Network (CNN) architecture used in the FashionMNIST experiments. (right) The reduced ResNet-18 architecture used for CIFAR-10, CIFAR-100, and tinyImageNet, is built using the BasicBlock(n_f, n_b, n_s) from (He et al., 2016), where n_f is the number of convolutional filters, n_b is the number of sub-blocks per block, and n_s is the stride of the layer.

ConvBlock	CNN	Reduced ResNet-18
Conv2D($n_{\text{in}}, n_{\text{out}}$)	ConvBlock($1, 32$)	BasicBlock(20, 2, 1)
ReLU	ConvBlock($32, 64$)	BasicBlock(40, 2, 2)
BatchNorm2D(n_{out})	Linear($64, c$)	BasicBlock(80, 2, 2)
Conv2D($n_{\text{out}}, n_{\text{out}}$)		BasicBlock(160, 2, 1)
ReLU		AveragePooling
BatchNorm2D(n_{out})		Linear($160, c$)
MaxPooling2D($2, 2$)		

A Reproducibility

In order to ensure that our work is not only reproducible but also easily accessible, we commit to open sourcing our code upon acceptance. Moreover, in order to increase the reproducibility of our results, wherever we present average performance over many runs, we do so in the form 95%-confidence intervals, in order for the numbers to also accurately reflect their inherent uncertainty.

B Architectures

Please refer to Table 6