Real-Time Evaluation in Online Continual Learning: A New Hope

Yasir Ghunaim[1*] Adel Bibi[2*] Kumail Alhamoud[1] Motasem Alfarra[1]
Hasan Abed Al Kader Hammoud[1] Ameya Prabhu[2] Philip H.S. Torr[2] Bernard Ghanem[1]

[1] King Abdullah University of Science and Technology (KAUST) [2] University of Oxford

Abstract

Current evaluations of Continual Learning (CL) methods typically assume that there is no constraint on training time and computation. This is an unrealistic assumption for any real-world setting, which motivates us to propose: a practical real-time evaluation of continual learning, in which the stream does not wait for the model to complete training before revealing the next data for predictions. To do this, we evaluate current CL methods with respect to their computational costs. We conduct extensive experiments on CLOC, a large-scale dataset containing 39 million time-stamped images with geolocation labels. We show that a simple baseline outperforms state-of-the-art CL methods under this evaluation, questioning the applicability of existing methods in realistic settings. In addition, we explore various CL components commonly used in the literature, including memory sampling strategies and regularization approaches. We find that all considered methods fail to be competitive against our simple baseline. This surprisingly suggests that the majority of existing CL literature is tailored to a specific class of streams that is not practical. We hope that the evaluation we provide will be the first step towards a paradigm shift to consider the computational cost in the development of online continual learning methods.

1. Introduction

Deep Neural Networks (DNNs) have demonstrated impressive success in solving complex tasks [20, 29, 42] when trained offline, for several passes, over large well-curated labeled datasets. However, in many real-world scenarios, data is only available in the form of a stream with a changing distribution. Due to this challenge, there has been a growing interest in the problem of learning from a time-varying stream, also known as Continual Learning (CL), which is a key challenge for DNNs due to a phenomenon known as *catastrophic forgetting* [17, 36]. In particular, when a DNN is trained with data from a new distribution, the DNN performance significantly drops on previously learned data.

* Equal Contribution.
Correspondence to: `yasir.ghunaim@kaust.edu.sa`
Code: github.com/Yasir-Ghunaim/RealtimeOCL

While mitigation efforts have been proposed, *e.g.* through regularizing the training [2, 25, 53], replaying previously seen examples [11, 23, 39], and many other approaches [16, 40, 51], current evaluation approaches are still far from real-world scenarios. For example, the majority of literature is on *Offline Continual Learning*, under which methods are allowed unlimited budget, both time and computation. Furthermore, the majority of CL evaluations are conducted on *small-scale* datasets with well-defined temporal distribution boundaries in the form of learning a sequence of tasks.

To that end, there has recently been a growing interest in the more realistic setting – *Online Continual Learning* (OCL). In such a setup [1, 4, 21, 32], CL methods are restricted to a single training pass over a shuffled split of existing offline CL benchmarks. This is certainly a step forward towards resolving some of the unrealistic assumptions of offline CL. However, current evaluations do not sufficiently address the challenges of real-time learning for high-throughput streams with rapid distribution changes.

To illustrate this, consider the problem of continuously learning a Twitter stream where 350K tweets are uploaded per minute on various trending topics [41]. Every uploaded tweet needs to be predicted with a DNN for misinformation and hate speech, among other things, while simultaneously learning and adapting to them. Given the scale at which data is being updated, there is an inherent key limitation on the time and computational budget affordable to learning incoming tweets, an aspect that is often overlooked in the prior art from the OCL literature. Consider an OCL method that is 10 times slower than the Twitter high throughput stream, *i.e.*, it takes 10 minutes to train on one minute worth of tweets (350K tweets). This inefficiency results in an accumulation of $\sim$3.1 million new samples that need to be predicted and trained on. Since it is not acceptable to pause

Figure 1. **OCL Real-Time Evaluation Example.** We show an example of real-time evaluation, using the CLOC dataset [7], of two different OCL methods $\mathcal{A}$ and $\mathcal{B}$. Method $\mathcal{B}$ is twice as slow as method $\mathcal{A}$. Both methods are evaluated on every incoming sample. Since $\mathcal{A}$ has a stream-model relative complexity of one, *i.e.* $\mathcal{C}_\mathcal{S}(\mathcal{A}) = 1$, it is able to train on all the stream samples. In contrast, $\mathcal{B}$, which has a relative complexity of two, requires two time steps to train on a single stream batch. Thus, $\mathcal{B}$ only trains on half of the stream samples.

all tweets from appearing online until the method training is complete, predictions for all new samples will be performed with an older version of the model. This poses a key challenge where efficient learning from streams becomes necessary. This is because slow-training OCL methods can result in subpar performance, as they resort to predicting new stream data using an older model. This behavior worsens for streams that experience a faster change in distribution.

In this paper, we propose a real-time evaluation protocol for OCL that factors in training computational complexity. Given a stream, consider an OCL method $\mathcal{A}$ that is as fast as the stream, *i.e.*, $\mathcal{A}$ can train on every step of revealed data before the stream presents new samples. Then, if an OCL $\mathcal{B}$ is twice as expensive as $\mathcal{A}$, then $\mathcal{B}$ will update the model for evaluation every other stream step, *i.e.*, the model will be updated half the number of times compared to $\mathcal{A}$. Figure 1 illustrates our proposed real-time evaluation. This is in contrast to all prior art [3, 4, 6] that (**1**) unreasonably allows an unlimited computational budget to train on any given stream data, and (**2**) unfairly compares OCL methods despite having different training complexity levels. Using our real-time evaluation protocol, we benchmark many existing OCL methods against a simple and inexpensive baseline, which mitigates forgetting by simply storing and replaying recently seen samples.

Contributions. We summarize our conclusions as follows: **(1)** We show that under our practical real-time evaluation, our simple baseline outperforms *all* the considered methods from the OCL literature, including recent SOTA approaches like ACE [6]. **(2)** We consider a complementary setup where the stream is as slow as the most training-expensive OCL method and compare that method against the compute-equivalent baseline. Under this computationally normalized setting, we find that the compute-equivalent baseline outperforms all existing methods. **(3)** Our experiments are consistent, holding for all the considered continual learning strategies, and extensive, amounting to more than 2 GPU-months. Our results highlight that the current progress in OCL needs to be rethought and a paradigm shift is needed. We hope our work will lead to a new direction for continual learning that takes into account the computational cost of each method.

2. Related Work

We briefly describe the current literature on offline and online continual learning. For a more comprehensive overview of the literature, we refer the reader to the detailed surveys by De Lange *et al.* [13] and Mai *et al.* [33].

Offline Continual Learning. Traditional continual learning strategies, which aim to mitigate forgetting, can be organized into three families of methods. *(i) Replay-based* methods store input samples, or alternatively learn to generate representative samples, while the model learns from the data stream. Later, the method retrains the model on the replay samples while the model learns from new data [32, 35, 45, 49, 50]. *(ii) Regularization* methods avoid the cost of storing samples and simply modify the model loss objective to regularize the training. While some of these methods penalize changes to important model parameters [2, 8, 25], other methods regularize the training by distilling knowledge from a model trained on past stream samples [18, 28, 46]. *(iii) Parameter isolation* methods train specific parameters for each task while freezing the parameters that are related to other tasks [22, 34, 40]. Despite the progress made by these methods, they assume an offline stream that allows many passes over each continual learning task. Concurrent work Prabhu *et al.* [37] alleviates this problem by imposing computational budget constraints and finds that simple methods based on experience replay outperform most prior continual learning works. In contrast, we study the more pragmatic setup, where the stream reveals data in real time.

Online Learning for Reduced Forgetting. OCL was defined with a protocol where training data is only seen once in a sequence of labeled tasks [32]. To reduce catastrophic forgetting, the field initially progressed towards better gradient-based constraints like GEM [32] and AGEM [10]. RWalk [8] quantified forgetting and provided a more efficient approach to limit changes in important model parameters. TinyER [11] rediscovered the effectiveness of experience replay, and HAL [9] enhanced it with learned anchors. However, this setup assumes the availability of an oracle during test time to determine which classifier head to use for inference [33, 48]. Additionally, the benchmarks in this setup often have large constraints on the replay buffer size [10, 11]. Due to these limitations, the class-incremental scenario started to gain attention, which is a more realistic setting for OCL.

In class-incremental online continual learning, benchmarks [3, 27] relaxed the need for task descriptors at test time, and introduced significantly larger buffer sizes in comparison to [10]. Multiple directions have emerged to tackle catastrophic forgetting in this setup. They can be classified into few groups. *(i) Regularization-based* approaches modify the classification objective to preserve previously learned representations or encourage more meaningful representations *e.g*. DER [5], ACE [6], and CoPE [14]. *(ii) Sampling-based* techniques focus on the optimal selection and storing of the most representative replay memory during online training, *e.g*. GSS [4], OCS [52], CBRS [12], CLIB [26], and InfoRS [47]. Alternatively, some sampling methods focus on better memory retrieval strategies that reduce forgetting, *e.g*. MIR [3], ASER [44], and GMED [24]. (iii) In other approaches, GDumb [38] proposed a degenerate solution to the problem of online learning ignoring the stream data and learning only on the memory samples. While these efforts have significantly advanced the field of OCL, they are mostly evaluated on benchmarks that do not reflect real-deployment conditions. First, these benchmarks heavily rely on artificially constructed small datasets with sudden shifts in classes. Second, these benchmarks are incapable of measuring whether models can rapidly adapt to new data under a fast-changing distribution shift, which is one of the main problems in classical online learning literature [43]. There have been efforts to address the second limitation by using new metrics that measure test accuracy more frequently during training [6, 26]. However, these metrics capture the adaptation to held-out test data rather than to the incoming future data. To remedy these limitations, there has been a new surge of benchmarks proposing datasets and evaluation protocols, which we discuss next.

Online Learning for Rapid Adaptation. Recent OCL benchmarks, *e.g*. CLOC [7] and CLEAR [30], introduce data ordered by timestamps, forming a temporal stream of evolving visual concepts over a long span of time. They demonstrate that their data has a natural distribution shift over time, requiring rapid adaptation to newer data. Additionally, they mimic the traditional online learning setup by measuring the capacity for rapid adaptation with the evaluation being done on future data from the stream. Our work extends the efforts in this direction. We adopt the CLOC benchmark and propose a more realistic real-time evaluation that encourages efficient learning. It is worth mentioning that [19] has explored certain aspects of real-time evaluation in continual learning. However, they focus on efficiency for embedded device deployment, rather than rapid adaptation. Moreover, while their work investigates the computational cost associated with each training method, they only report the cost as an evaluation metric. In contrast, we constrain the training procedure by each method's computational cost.

3. Methodology

We start with the classical problem statement for online continual learning. Then, we formally introduce our proposed real-time evaluation that factors in training complexity through delayed evaluations.

3.1. Online Continual Learning

Online continual learning [7, 43] is the problem of learning a θ parameterized function $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ that predicts a label $y \in \mathcal{Y}$ for an input image $x \in \mathcal{X}$. Unlike, classical supervised learning, the learning is performed on a distribution-varying stream $\mathcal{S}$ revealing data sequentially over steps $t \in \{1, 2, \dots, \infty\}$. In particular, at every step t,

1. $\mathcal{S}$ reveals a set of images $\{x_i^t\}_{i=1}^{n_t} \sim \mathcal{D}_{j\leq t}$;
2. f_{θ_t} generates predictions $\{\tilde{y}_i^t\}_{i=1}^{n_t}$ for $\{x_i^t\}_{i=1}^{n_t}$;
3. $\mathcal{S}$ reveals true labels $\{y_i^t\}_{i=1}^{t}$;
4. f_{θ_t} is evaluated by comparing $\{\tilde{y}_i^t\}_{i=1}^{n_t}$ to $\{y_i^t\}_{i=1}^{n_t}$;
5. A learning method trains f_{θ_t}, a criterion computes the training loss, then the parameters are updated to θ_{t+1}.

Note that $\mathcal{D}_{j\leq t}$ denotes a varying distribution that might not necessarily need to change at every stream step t. For example, if at step $t = 5$ we have $j = 1$, this means the revealed data over all five previous steps is sampled from the

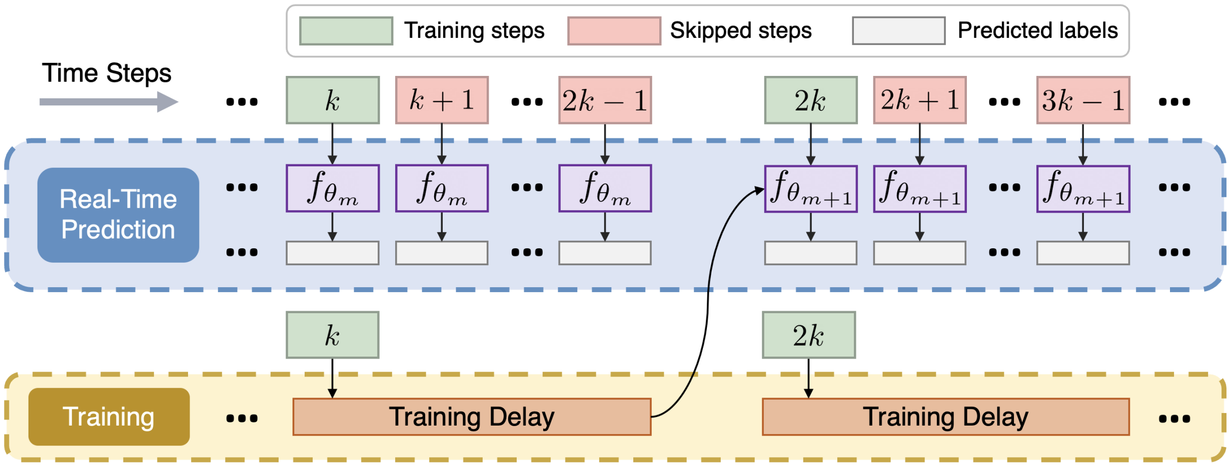

Figure 2. **OCL Real-Time Evaluation Setup.** In OCL, models perform real-time prediction on each revealed sample from the stream. At the same time, model training is performed on the revealed samples dictated by the method's delay. In this example, we demonstrate the evaluation procedure when the stream-model complexity k is larger than 1. Due to the mismatch between the stream speed and the model computational cost, an "older version" of the model predicts samples while the model is being trained. Therefore, k-1 potential training batches are skipped for each training step.

same distribution $\mathcal{D}_1$. Moreover, observe that, unlike offline continual learning, online continual learning captures the capacity to adapt to new data since the prediction is performed before the training.

Key Issues. As described earlier, the OCL evaluation in the literature [7, 43] overlooks the training complexity in step (5). Under such a setup, all OCL methods have their parameters updated to θ_{t+1} before the stream $\mathcal{S}$ reveals images of the next step. Therefore, different OCL methods are evaluated irrespective of the training time and computational complexity at step (5). This is equivalent to evaluating different OCL methods on streams that present data at different rates. In real-time settings, the stream reveals data at a rate that is independent of the training complexity of the OCL method. To that end, we propose a real-time evaluation paradigm for OCL that factors in training complexity through delayed evaluations.

3.2. Real-Time Evaluation for OCL

As mentioned earlier, we need to define a notion of a fixed stream rate. That is to say, the stream reveals data at a fixed rate irrespective of how long OCL methods take to update the model in step (5). For streams that are twice as fast as it takes to train f_θ, OCL methods will have their model updated on only half the number of revealed sets of images from the stream. Note that the latest updated version of f_θ will continue to provide predictions for newly revealed images during training even for those images the model did not train on. This setting reflects the realistic scenario of a server receiving many queries uploaded by users, where each query needs a prediction irrespective of whether an in-house model is being trained. Given a stream $\mathcal{S}$ and an OCL method $\mathcal{A}$, we first define the notion of stream-model relative complexity $\mathcal{C}_\mathcal{S}(\mathcal{A}) \in \mathbb{R}^+$. In particular, for a stream-model relative complexity $\mathcal{C}_\mathcal{S}(\mathcal{A}) = 1$, the continual learning method $\mathcal{A}$ can update the model θ in step (5) before the stream $\mathcal{S}$ reveals the data of the next step. For any stream-model relative complexity $\mathcal{C}_\mathcal{A}(\mathcal{A}) = k > 1$, the stream is k-faster than the continual learning method $\mathcal{A}$. With this notation, we now propose our modified version of the OCL setting along with the corresponding real-time evaluation. Given a model f_{θ_m}, a stream $\mathcal{S}$, and an OCL method $\mathcal{A}$ with $\mathcal{C}_\mathcal{S}(\mathcal{A}) = k$, at every step t,

1. $\mathcal{S}$ reveals a set of images $\{x_i^t\}_{i=1}^{n_t} \sim \mathcal{D}_{j\leq t}$;
2. f_{θ_m} generates predictions $\{\tilde{y}_i^t\}_{i=1}^{n_t}$ for $\{x_i^t\}_{i=1}^{n_t}$;
3. $\mathcal{S}$ reveals true labels $\{y_i^t\}_{i=1}^{t}$;
4. f_{θ_m} is evaluated by comparing $\{\tilde{y}_i^t\}_{i=1}^{n_t}$ to $\{y_i^t\}_{i=1}^{n_t}$;
5. If $\text{mod}(t-1, k) = 0$, then the continual learner $\mathcal{A}$ completes updating $f_{\theta_m} \leftarrow f_{\theta_{m+1}}$ and a new instance of training on f_{θ_m} commences (see Figure 2).

In this setting, OCL methods that are computationally more expensive to train will be updated fewer times. Therefore, for streams $\mathcal{S}$ where the distribution $\mathcal{D}_j$ changes rapidly (perhaps as often as every stream step), OCL methods with a large stream-model relative complexity may produce erroneous predictions, since the models are not updated enough and cannot adapt to the distribution change.

On the Computation of $\mathcal{C}_\mathcal{S}$. Since $\mathcal{C}_\mathcal{S}$ only measures the relative complexity between the stream and an underlying continual learning method, we first introduce a minimal inexpensive OCL method as a baseline (denoted $\mathcal{A}$). For ease of comparison of the considered methods, we then assume that, due to the inexpensive nature of $\mathcal{A}$, online continual learning will be as fast as the stream where the stream-model relative complexity $\mathcal{C}_{\mathcal{S}} = 1$. In particular, we consider a baseline that mitigates forgetting by simply storing and replaying recently seen samples. Then, given any other OCL method $\mathcal{B}$, we use the relative training FLOPs between $\mathcal{B}$ and $\mathcal{A}$ to determine $\mathcal{C}_{\mathcal{S}}(\mathcal{B})$. For example, ACE [6] only modifies the loss objective of the baseline, and thus it is equivalent to the baseline in computational complexity. On the other hand, PoLRS [7] maintains and performs operations on three copies of the model. These copies are trained on every incoming batch, making PoLRS require $3\times$ the FLOPs needed by the baseline. As noted earlier, a method of a stream-model relative complexity value of 3 performs the update in step (5) once every three stream steps. This corresponds to a delay in model updates by two steps. In Table 1, we summarize the corresponding delay of several popular OCL methods in our real-time evaluation setup.

Fair Comparison of OCL Methods with Different $\mathcal{C}_{\mathcal{S}}$. We proposed a realistic setup of evaluating continual learners based on their training complexity with respect to the stream speed. However, one might argue that more expensive training routines could be accelerated by deploying more computational hardware allowing them to train on each revealed sample from the stream. For example, while PoLRS [7] requires $3\times$ more FLOPs than the simple baseline, one could deploy $3\times$ more hardware to make $\mathcal{C}(\mathcal{A}) = 1$ for PoLRS. This setup mimics the scenario of having a slow stream that matches the speed of the more expensive training method.

While the aforementioned setup normalizes the computational requirement of a given learning method $\mathcal{A}$ to match the stream speed, one should allow simpler training methods the same computational complexity. To that end, we propose boosting simpler and faster training methods to match the computational requirements of more complex methods. For example, we boost the simple experience replay method by sampling a larger number of instances from the replay buffer at each step t to match the FLOPs of other training schemes. We note here that although this modification is both simple and naive, we found empirically that it is sufficient to outperform *all* considered OCL methods. We leave the rest of the implementation details to Section 4.2.

4. Experiments

We compare OCL methods under our proposed real-time evaluation setup given two stream speeds: fast and slow, which capture different application scenarios. We benchmark the majority of OCL methods, which we group into three categories. (1) *Regularization-based* OCL methods regularize training by modifying the classification loss. For example, RWalk [8] uses a regularizer to penalize abrupt changes in the parameters of the model. ACE [6] introduces an asymmetric loss to treat memory buffer samples differently from incoming stream samples, while LwF [28] distills knowledge from previous steps. (2) *Learning rate scheduler* methods, in this case PoLRS [7], dynamically adapt the learning rate to changes in the stream distribution. (3) *Sampling-based* methods alter the strategy used to update the memory buffer, *e.g*. GSS [4], or change the memory retrieval strategy, *e.g*. MIR [3].

Table 1. **Training complexity and delay of considered OCL methods.** *Note that GSS has a complexity of 6.5 but we rounded it down to 6 to facilitate the setup. This rounding may give GSS a slight advantage, but nonethless, it is outperformed by the baseline, ER, in all experiments.

CL Strategy	**Method**$(\mathcal{A})$	$\mathcal{C}_{\mathcal{S}}(\mathcal{A})$	**Delay**
Experience Replay	ER [11]	1	0
Regularizations	ACE [6]	1	0
	LwF [28]	4/3	1/3
	RWalk [8]	2	1
LR Scheduler	PoLRS [7]	3	2
Sampling Strategies	MIR [3]	5/2	3/2
	GSS [4]	6*	5

Datasets. We use the large-scale online continual learning dataset CLOC [7], which contains 39 million time-stamped images exhibiting natural distribution shifts. The task is to identify the geolocation of a given image where the total number of geolocation labels is 712. To ensure consistency with CLOC [7], we adopt the same dataset split approach. Specifically, we use the first 5% of the stream for hyperparameter selection, uniformly sample 1% from the remaining stream to build a held-out set for measuring backward/forward transfer (reported in the appendix), and utilize the rest of the stream for training and online evaluation. Similar to CLOC [7], we compare OCL methods using the *Average Online Accuracy* metric, which measures the ability of models to adapt to incoming stream samples[1].

Implementation Details. At each step t during the experiments, $\mathcal{S}$ reveals a set of 128 images, and OCL methods augment the training batch with another 128 images sampled from a memory buffer. This routine is performed until a single pass over the stream is completed. We use ImageNet [15] pre-trained ResNet50 [7, 20] as a backbone. We use SGD with a learning rate of 5×10^{-3} for all OCL methods except for PoLRS [7], which works best with a learning rate of 1×10^{-3}. Unless otherwise stated, we set the size of the replay buffer to 4×10^4 for all considered methods. We

[1] This metric should not be mistaken for a training accuracy, as it evaluates the model on the next *unseen* training batch before the batch is used for model training.

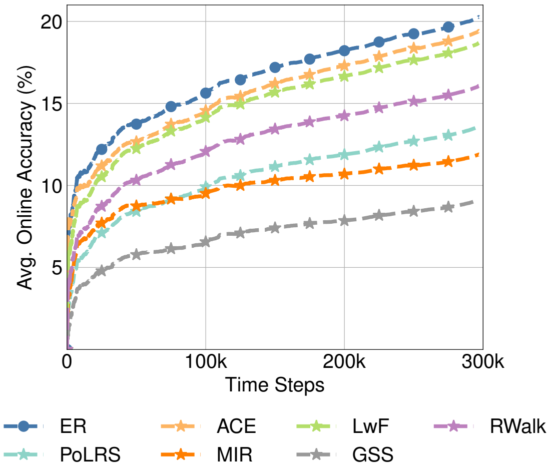

Figure 3. **Fast Stream Evaluation.** We compare the Average Online Accuracy of six methods from the OCL literature evaluated under the fast-stream setting. We observe that our inexpensive baseline, ER, outperforms all the considered methods. Surprisingly, the most computationally complex methods, MIR, PoLRS, and GSS, underperform the more efficient methods.

use the same hyperparameters reported in all corresponding papers with the exception of the regularization parameter for RWalk [8]. We find that their reported regularization $\lambda = 0.1$ is too small for the CLOC dataset; we cross validate it and find that $\lambda = 2$ works best. Our implementation extends the codebase of Avalanche [31], which is a popular continual learning framework. We leave all the remaining implementation details and ablations to the appendix.

4.1. Fast Stream: Real-Time Evaluation

We first consider the fast-stream scenario, where the stream reveals data at a high rate. We consider the simple baseline *ER* [11], which performs one gradient step on a batch of 256 (128 revealed by the stream and 128 from memory) images at every stream step. The 128 images from memory are sampled uniformly. For simplicity and ease of comparison, we assume that *ER* has a stream-model relative complexity of $\mathcal{C}_\mathcal{S}(ER) = 1$. Consequently, any OCL method that is computationally more expensive than *ER* will not keep up with the stream speed, and thus will be forced to skip training on a fraction of the incoming stream samples.

Benchmarking Relative Complexity of OCL Methods. As discussed earlier, we use the total number of FLOPs required for each method to train on a stream as a proxy for the stream-model relative complexity $\mathcal{C}_\mathcal{S}$. We first compute the total FLOPs required to train *ER* on a small subset of CLOC composed of 10^3 images with a batch size of 10. We then compute the corresponding FLOPs for various OCL methods and normalize them by the FLOPs of *ER*. Since the backbone is shared across all methods, we further verify this normalized complexity by counting the effective number of forward and backward passes of each method. Because we assume that *ER* is effectively as fast as the stream, methods that are twice more expensive in FLOPs have a complexity $\mathcal{C}_\mathcal{S}$ of 2. We report in Table 1 the stream-model relative complexity for OCL methods compared against *ER*. Note that PoLRS is 3 times as expensive as *ER* since it requires training 3 different models to update the learning rate. In addition to the 128 images revealed by the stream and the 128 images sampled from memory at step t, GSS performs additional forward and backward passes on 10×128 memory samples to ensure diversity when updating the memory buffer [4]. Therefore, GSS is roughly 6 times more expensive than *ER*. For each method in Table 1, we report the corresponding delay in stream steps, which is a consequence of some methods being slower than the stream. A delay of 2 means that the stream will reveal 2 steps worth of images before the model is updated.

Effect of Training Efficiency. We start our analysis by investigating the effect of training efficiency on the performance of different OCL methods under the real-time evaluation setup. We plot the Average Online Accuracy curves per step in Figure 3, where the simple *ER* baseline is in blue. Each method is evaluated according to its corresponding training delay reported in Table 1. Surprisingly, *ER* outperforms all considered methods, and in some cases, by large margins. Interestingly, the most computationally intensive methods, MIR, PoLRS, and GSS, exhibit the lowest performance when real-time evaluation is considered. Specifically, the performance gap at the end of the stream reaches 11%. In fact, we note that the Average Online Accuracy values of all considered methods, irrespective of how recently they were introduced to the literature, are approximately ordered based on their stream-model relative complexity $\mathcal{C}_\mathcal{S}$; the larger $\mathcal{C}_\mathcal{S}$ is, the worse the performance is under real-time evaluation. We attribute this trend to the fact that these inefficient methods may not be capable of keeping up with the fast-changing distribution $\mathcal{D}_j$ in the stream when they are subjected to a higher delay. For example, although PoLRS was proposed and tailored for the recent CLOC benchmark, it is significantly outperformed by the older method LwF when evaluated under the real-time paradigm. This hints that computational complexity can be a crutch for OCL methods learning on fast streams. Our results show that while the current literature aims to improve learning by leveraging more expensive and advanced learning routines, the penalty incurred by delay overshadows any potential improvements gained by the algorithmic contributions. Notably, the state-of-the-art ACE method, which is as efficient as *ER* and thus evaluated under no training delay, performs almost as well as the baseline. Additionally, ACE outperforms more expensive methods when evaluated on small-scale datasets (refer to Small-Scale Experiments in the appendix). Overall, our findings suggest that practical

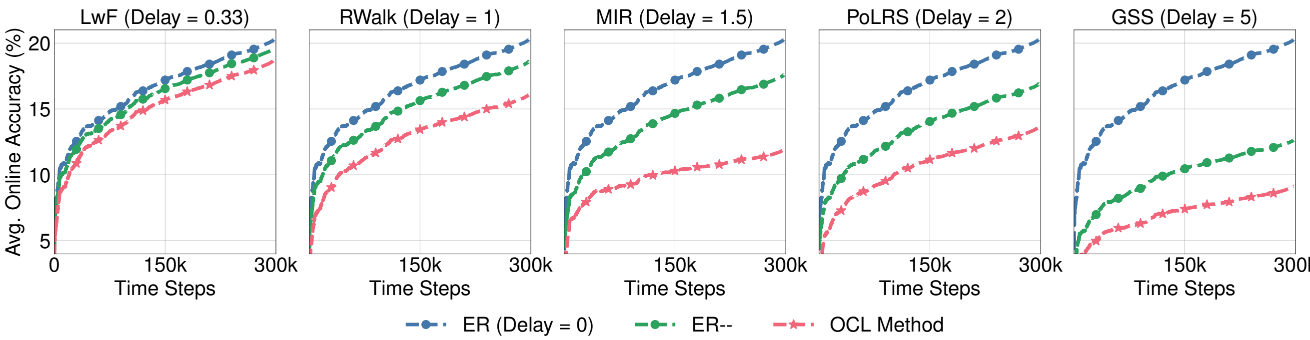

Figure 4. **Fast Stream - Training Data Normalization.** We compare each method against *ER* and its delayed version *ER*--. *ER*-- performs extra gradient steps per time step to match the delay of the compared-against method, so it trains on fewer samples than *ER*. We highlight that even when *ER*-- is unfairly delayed to match each OCL method's delay, it outperforms all considered methods.

OCL methods, deployed in real-time applications, should prioritize efficient learning over expensive optimization.

Training Data Normalization. Previously, we have shown that delayed evaluation leads to larger performance degradation as the delay gets longer. However, this could be attributed to the fact that OCL methods with larger delays end up effectively training on a fewer number of samples. This raises the questions: *Do computationally more expensive methods with delays perform worse than* ER *because they cannot cope with the changing distribution in the stream?*, or *is the performance degradation due to expensive OCL methods being trained on effectively a fewer number of training examples?* To address these questions, we conduct pairwise comparisons between *ER* and each previously reported OCL method. Assuming that the stream speed is fixed, we modify *ER* to match the computational expense of the respective method by training on the same amount of data for a longer duration. To achieve this computational matching, we subject *ER* to a delay by performing additional gradient descent steps at each stream step. We refer to this modified baseline as *ER*--. The number of additional updates in *ER*-- is determined such that its delay matches the delay of the corresponding OCL method. This guarantees that *ER*-- trains on the same number of training examples compared to other OCL methods.

Figure 4 shows comparisons of *ER* and *ER*-- to each of the considered OCL methods. Since the non-delayed *ER* is more efficient than *ER*--, *ER* consistently outperforms *ER*--, confirming that efficiency is key for the real-time evaluation of OCL. More interestingly, even though *ER*-- matches the complexity of each compared-against method and is subject to the same delay, it still outperforms all considered OCL methods. Moreover, while the gap between *ER*-- and expensive approaches, *e.g*. GSS, is smaller than the gap to *ER*, expensive methods still lag behind *ER*-- by up to 3.5%. This demonstrates that the degraded performance of considered OCL methods is not due to the fewer number of observed training examples. On the contrary, expensive OCL methods seem to be unable to cope with distribution-changing streams. Under real-time evaluations of fast streams, simple methods such as *ER* and *ER*-- may be more suitable for real-world deployment.

4.2. Slow Stream: Complexity Normalization

In the fast stream setting, we considered the scenario where the stream is as fast as the *ER* baseline, *i.e*., $\mathcal{C}_\mathcal{S}(ER) = 1$. In this section, we consider streams that are as slow as the more expensive OCL methods. This setup motivates the following question: *How do existing OCL methods perform if they were deployed on streams that match their training complexity?* Slower streams may enable expensive methods to train on the entire stream, without having to skip any stream steps. *Would previously considered methods, which struggled in the fast-stream setting, be suitable for OCL deployment in this slow-stream scenario?* We compare various OCL methods *under no delay* against their corresponding modified baseline *ER*++, which computes additional number of gradient steps per batch to match the complexity of the compared-against method, as determined from Table 1. This mimics the comparison on slow stream speeds, where no stream steps are skipped and comparisons are performed under *normalized complexity*. Note that ACE only modifies the loss objective of *ER*, so it matches the complexity of the baseline $\mathcal{C}_\mathcal{S}(ACE) = 1$. As a result, evaluating ACE in the slow-stream setting is identical to evaluating it in the fast-stream setting, which was already done in Figure 3. Therefore, we do not compare to ACE again in the following experiments.

We report the comparisons in Figure 5, where *ER*++ is shown in green and OCL methods in red. Moreover, we show *ER*, which is unfavorably compared against both, in blue. While RWalk outperforms *ER* in the slow stream setting, this is an unfair comparison, since *ER* is not utilizing the fact that the stream is slow and that it can benefit from further training. Once we simply add a few iterations to *ER*, *i.e*., *ER*++, so as to match the training complexity of

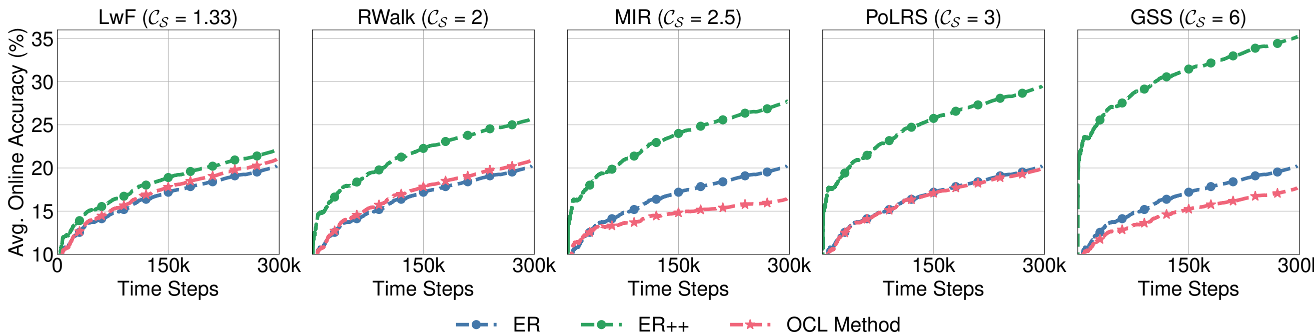

Figure 5. **Slow Stream.** We compare each method against *ER* and *ER++*, which performs extra gradient steps per time step to match the complexity $\mathcal{C}_S$ of the compared-against method. *ER++* outperforms all the considered methods, sometimes by large margins.

RWalk, *ER++* performs far better than RWalk. We find this to be consistent across all methods.

Interestingly, computationally intensive methods, *e.g*., GSS and MIR, perform worse even when compared to *ER*, which does not exploit the fact that the stream is slow. *ER++* is significantly better than GSS and MIR, by 17% and 11% respectively, expanding the performance gap to the baseline by around 15% and 7%. These results question the suitability of the previously proposed methods to realistic OCL setups with large-scale datasets. Moreover, our results suggest that a computationally unfair evaluation setup (*i.e*. not normalizing training complexity) in slow stream OCL can lead to misleading performance conclusions, as is the case with RWalk. We highlight that PoLRS was originally tested on a slightly different setup to the slow stream, which included extra information represented by a user album [7]. We find that when the album information is removed and their reported baseline, *i.e*., *ER*, is tuned properly, PoLRS has a similar performance to *ER* as shown in Figure 5. This is still surprising since PoLRS is three times more expensive than *ER*. More surprisingly, when matching the training complexity of *ER* to PoLRS on the slow stream, *ER++* outperforms PoLRS by 9.5%. These results on the slow-stream setting support our earlier observation that current OCL methods are not yet suited for practical deployment.

4.3. Effect of Memory Size

We conduct experiments to test whether existing methods can outperform *ER* under different memory buffer sizes. We repeat the fast-stream experiment from Section 4.1 with memory budgets of 1×10^4, 2×10^4, and 4×10^4 samples. The results are summarized in Table 2. A larger memory size leads to a higher Average Online Accuracy across all methods. However, we emphasize that the trend across methods aligns with our previous finding that expensive-training OCL methods perform worse than their less expensive counterparts. This is evident from the ordering of the methods in Figure 3, which holds regardless of the memory budget. Notably, *ER* outperforms all considered methods.

Table 2. **Memory Budget Analysis.** We test the effect of varying the memory budget in the fast-stream setting. We observe that 1) increasing the memory size results in better performance regardless of the method, and 2) ER outperforms all the considered methods regardless of the memory budget.

Method	Memory Size		
	1×10^4	2×10^4	4×10^4
ER [11]	**18.09**	**19.37**	**20.27**
ACE [6]	17.53	18.55	19.42
LwF [28]	16.87	17.87	18.69
RWalk [8]	14.60	15.29	16.07
PoLRS [7]	12.40	12.98	13.59
MIR [3]	11.37	11.64	11.89
GSS [4]	8.01	8.59	9.10

5. Conclusion

We proposed a real-time evaluation benchmark for online continual learning, including a fast stream that continues to reveal new data even when the model has not completed training. This realistic setup forces computationally intensive methods to skip more training samples, highlighting the differences in training complexity between the considered methods. Our results show that, under this setting, all considered methods underperform the simple *ER* baseline. We also explored scenarios where we normalized the number of seen training data or the computational complexity per time step, leading to the same conclusion that current methods are not yet optimized for real-world deployment.

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-CRG2021-4648, SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence (SDAIA-KAUST AI), Saudi Aramco, and UKRI grant Turing AI Fellowship EP/W002981/1. We thank the Royal Academy of Engineering and FiveAI for their support. Ameya Prabhu is funded by Meta AI Grant No DFR05540.

References

[1] Motasem Alfarra, Zhipeng Cai, Adel Bibi, Bernard Ghanem, and Matthias Müller. Simcs: Simulation for online domain-incremental continual segmentation. *arXiv preprint arXiv:2211.16234*, 2022. 1

[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *European Conference on Computer Vision (ECCV)*, 2018. 1, 2

[3] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 2, 3, 5, 8

[4] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. 1, 2, 3, 5, 6, 8

[5] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 3

[6] 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 (ICLR)*, 2022. 2, 3, 5, 8

[7] Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *International Conference on Computer Vision (ICCV)*, 2021. 2, 3, 4, 5, 8

[8] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *European Conference on Computer Vision (ECCV)*, 2018. 2, 3, 5, 6, 8

[9] Arslan Chaudhry, Albert Gordo, David Lopez-Paz, Puneet K. Dokania, and Philip Torr. Using hindsight to anchor past knowledge in continual learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021. 3

[10] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations (ICLR)*, 2019. 3

[11] 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 *International Conference on Machine Learning (ICML)*, 2019. 1, 3, 5, 6, 8

[12] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *International Conference on Machine Learning (ICML)*, 2020. 3

[13] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE TPAMI)*, 2021. 2

[14] Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *International Conference on Computer Vision (ICCV)*, 2021. 3

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 5

[16] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017. 1

[17] Robert M. French. Catastrophic forgetting in connectionist networks, 1999. 1

[18] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-dfcil: Relation-guided representation learning for data-free class incremental learning. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[19] Tyler L Hayes and Christopher Kanan. Online continual learning for embedded devices. In *Conference on Lifelong Learning Agents*, 2022. 3

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 5

[21] Huiyi Hu, Ang Li, Daniele Calandriello, and Dilan Gorur. One pass imagenet. In *Conference on Neural Information Processing Systems Workshops (NeurIPSW)*, 2021. 1

[22] Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao Tao, Dongyan Zhao, Jinwen Ma, and Rui Yan. Overcoming catastrophic forgetting for continual learning via model adaptation. In *International Conference on Learning Representations (ICLR)*, 2019. 3

[23] David Isele and Akansel Cosgun. Selective experience replay for lifelong learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018. 1

[24] Xisen Jin, Junyi Du, and Xiang Ren. Gradient based memory editing for task-free continual learning. 2021. 3

[25] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. In *Proceedings of the National Academy of Sciences (PNAS)*, 2017. 1, 2

[26] Hyunseo Koh, Dahyun Kim, Jung-Woo Ha, and Jonghyun Choi. Online continual learning on class incremental blurry task configuration with anytime inference. In *International Conference on Learning Representations (ICLR)*, 2022. 3

[27] 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 (ICLR)*, 2020. 3

[28] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (IEEE TPAMI)*, 2017. 2, 5, 8

[29] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[30] Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 3

[31] Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido M. Van De Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German I. Parisi, Fabio Cuzzolin, Andreas S. Tolias, Simone Scardapane, Luca Antiga, Subutai Ahmad, Adrian Popescu, Christopher Kanan, Joost Van De Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. Avalanche: An end-to-end library for continual learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 6

[32] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 1, 2, 3

[33] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 2022. 2, 3

[34] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[35] Andrea Maracani, Umberto Michieli, Marco Toldo, and Pietro Zanuttigh. Recall: Replay-based continual learning in semantic segmentation. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[36] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. 1989. 1

[37] Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet Dokania, Philip HS Torr, Ser-Nam Lim, Bernard Ghanem, and Adel Bibi. Computationally budgeted continual learning: What does matter? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[38] 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 (ECCV)*, 2020. 3

[39] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[40] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 1, 3

[41] David Sayce. The number of tweets per day in 2022, Aug 2022. 1

[42] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 2015. 1

[43] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 2011. 3, 4

[44] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. Online class-incremental continual learning with adversarial shapley value. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021. 3

[45] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 2

[46] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[47] Shengyang Sun, Daniele Calandriello, Huiyi Hu, Ang Li, and Michalis Titsias. Information-theoretic online memory selection for continual learning. In *International Conference on Learning Representations (ICLR)*, 2022. 3

[48] Gido M. van de Ven and Andreas S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 3

[49] Andrés Villa, Juan León Alcázar, Motasem Alfarra, Kumail Alhamoud, Julio Hurtado, Fabian Caba Heilbron, Alvaro Soto, and Bernard Ghanem. Pivot: Prompting for video continual learning. *arXiv preprint arXiv:2212.04842*, 2022. 2

[50] Andrés Villa, Kumail Alhamoud, Victor Escorcia, Fabian Caba, Juan León Alcázar, and Bernard Ghanem. vclimb: A novel video class incremental learning benchmark. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[51] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1

[52] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. In *International Conference on Learning Representations (ICLR)*, 2022. 3

[53] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, 2017. 1