Computationally Efficient Rehearsal for Online Continual Learning

Charalampos Davalas, Dimitrios Michail, Christos Diou, Iraklis Varlamis, and Konstantinos Tserpes

Harokopio University of Athens, Department of Informatics and Telematics, Athens, Greece, 17778

Abstract. Continual learning is a crucial ability for learning systems that have to adapt to changing data distributions, without reducing their performance in what they have already learned. Rehearsal methods offer a simple countermeasure to help avoid this catastrophic forgetting which frequently occurs in dynamic situations and is a major limitation of machine learning models. These methods continuously train neural networks using a mix of data both from the stream and from a rehearsal buffer, which maintains past training samples. Although the rehearsal approach is reasonable and simple to implement, its effectiveness and efficiency is significantly affected by several hyperparameters such as the number of training iterations performed at each step, the choice of learning rate, and the choice on whether to retrain the agent at each step. These options are especially important in resource-constrained environments commonly found in online continual learning for image analysis. This work evaluates several rehearsal training strategies for continual online learning and proposes the combined use of a drift detector that decides on (a) when to train using data from the buffer and the online stream, and (b) how to train, based on a combination of heuristics. Experiments on the MNIST and CIFAR-10 image classification datasets demonstrate the effectiveness of the proposed approach over baseline training strategies at a fraction of the computational cost.

Keywords: Catastrophic forgetting · Continual learning · Online learning.

1 Introduction

Continual learning aims at developing methods for adapting to new data distributions without dropping their performance on previously learned tasks. Decrease of performance on previous tasks, also known as *catastrophic forgetting*, occurs due to the fact that data is presented to the model incrementally and drawn from different distributions, essentially violating the i.i.d. assumption [1]. Several methods have been proposed to address the catastrophic forgetting problem, including (a) regularization methods [9] that pose constraints on the parameter update mechanism during training, (b) rehearsal/replay methods [14], that keep a set of representative past samples to be used along with new data in

training, and (c) parameter isolation methods that train only a subset of model parameters with the new data [10]. Despite their relative simplicity, rehearsal methods seem to work surprisingly well, as shown in a recent survey [3]. Most methods presented in the bibliography capitalize on "offline" (batch) learning, in which case all the data for all tasks are available for training in each iteration. Online continual learning introduces several additional challenges, including the continuously changing effectiveness in the current task, the forgetting of previous tasks, and the computational complexity of the training procedure. Edge and/or real-time application environments, can impose significant restrictions in the available computation resources for model training [4, 12, 13]. In this paper, we consider online continual learning for image classification under the prism of computational efficiency. For this purpose, we explore training strategies that can be applied at each training step, and balance the trade-off between model effectiveness and computational complexity. A rehearsal-based online continual learning setup is used to evaluate several training strategies, which involve decisions on *when* and *how* to train.

2 Related Work

Rehearsal-based methods in continual learning [3] mix selected samples from previous tasks with samples from new tasks during training. In task incremental continual learning, we assume clearly divided task boundaries, with all data of each task provided incrementally, i.e. all training data from the first task, then all data from the second task etc. [3, 14]. A way out of this relaxation to the more general problem of online continual learning may be offered in the approach of [11], where a Bayesian approach to infer the task context is suggested. Another approach by [17] presents an algorithm that uses the Shannon entropy as a measure to select task samples that are representative of previously seen classes without being affected by the fact that task boundaries are unknown. One prominent method for the Incremental Learning in Online Scenario has recently been introduced in [5]. This method combines various techniques to avoid catastrophic forgetting, including an adaptation of iCaRL [14] to the online learning case. A common attribute of these incremental learning methods is that they use a "static" training strategy. The training procedure takes place at predefined intervals (e.g., after a fixed number of samples is observed), while the number of training iterations/epochs at each training step is also fixed. In many occasions, this can lead to unnecessary training iterations, which add to the time complexity and can lead to overfitting. Our work is targeting the general online continual learning setting, where the task boundaries in the online stream of input images are not known in advance. The timeliness and memory objectives are considered together with complexity, plasticity, scalability and accuracy. Those characteristics allow our work to depart from previous research and deliver the following contributions:

– We propose a *decision mechanism* for determining *when* to train. This mechanism offers a significant advantage according to the time complexity. An-

other advantage is that this mechanism is suitable for detecting any changes in concept, therefore dealing with the issue of task agnosticism.
– We propose a *dynamic rehearsal strategy* as a solution of *how* to train, in an unpredictable, online setup. We propose strategies for dynamically determining the number of training iterations and learning rate based on the error of the model, as well as based on the convergence of the model parameters.

3 Online Continual Learning

3.1 Scenario

In a real-world setting, boundaries between different image classification problems, are not known in advance. We consider the following motivating scenario, where a constant stream of annotated data is used for model training in an online fashion as shown in Figure 1. The stream of data is non-i.i.d., since it is sampled from different tasks in each time period. The task boundaries and identities are unknown. Each task is a sequence of annotated samples from a set of n classes $C_1, \ldots, C_n$. For simplicity, we assume that data comes in fixed-size batches (e.g. 32 samples per batch), so the stream is considered as a sequence of batches, which comprises of a set of samples $B_t = (X_t, Y_t)$, with $t \geq 1$ representing the batch index. The goal is to train a sequence of models, where each model h_t for $t \geq 1$ is trained with B_t in an online manner.

3.2 The proposed online rehearsal method framework

This section describes the general rehearsal framework, which is more formally defined in Algorithm 1. During training, we maintain two buffers P and R, called the *Postponed* and *Rehearsal* buffers respectively. Let's assume that at time-step t a batch of new samples $B_t = (X_t, Y_t)$ is acquired from the stream, h_{t-1} is the model from the previous time-step, and P_t and R_t are the current state of the two buffers. The Postponed buffer is initially empty and the Rehearsal buffer initially contains a user-selected fixed number $q = |R|/n$, where n is the number of classes, of exemplars per class sampled uniformly at random. Algorithm 2 covers a wide range of training strategies for the online continual learning scenario.

In the one extreme the model is trained in each iteration by stacking the new batch B_t with a single (sampled uniformly at random) batch from the Rehearsal buffer [2]. In the other extreme, we may assume that all task data is available at a single moment in time, and an oracle correctly predicts the task boundaries and retrains at the end of each task using all the data. Between these two extreme cases, without any knowledge about the task boundaries, we are forced to deal with several issues such as: (a) when to train, (b) how to mix samples from the Postponed and Rehearsal buffers, (c) how many iterations to train at the current time-step, and (d) what learning rate schedule to use. At each time-step the algorithm first decides whether to train the model with a new batch or not. If not, the new batch is appended to the Postponed buffer P_t and

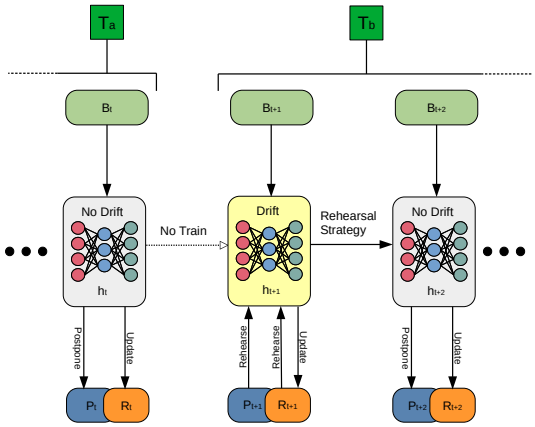

Fig. 1. Proposed strategy outline. Each batch $B_t, B_{t+1}, B_{t+2}, \ldots$ belongs to a specific classification task $T_a, T_b, \ldots$ and $h_t, h_{t+1} \ldots$ is the sequence of produced models after each step. A new batch B_t is added to a Postponed buffer P_t if the model doesn't need any updates. When training is needed in the next batch (i.e B_{t+1}), both the new Rehearsal buffer R_{t+1} and a new Postponed buffer P_{t+1} are mixed together with B_{t+1} for rehearsal training. In every step the Rehearsal buffer is being updated, regardless of training schedule.

the algorithm returns. If the algorithm decides to train the model, it employs a *rehearsal strategy* (see Section 4 for the supported options). Part of the strategy is to create mini-batches by combining new samples (i.e. from $P_t \cup B_t$) with samples from the Rehearsal buffer R_t. If training took place in the previous time-step, then $P_t = \emptyset$. After each training step, the Rehearsal buffer R_t is updated (as explained in the following) and the Postponed buffer P_t is cleared.

Algorithm 1: Streaming rehearsal framework single timestep t.

Input: Model h_t, State S_t (global variables from previous timesteps), Postponed buffer P_t, Rehearsal buffer R_t, New batch $B_t = (X_t, Y_t)$ from stream S

Output: updated h'_t, S'_t, P'_t, and R'_t

$train \leftarrow decideToTrain(\{h_t, S_t, P_t \cup B_t, R_t\})$;
if $train = False$ **then**
  update $P'_t \leftarrow P_t \cup B_t$ and state S'_t;
  return $\{h_t, S'_t, P'_t, R_t\}$;
$h'_t, S'_t \leftarrow RehearsalTraining(\{h_t, S_t, P_t \cup B_t, R_t\})$;
$R'_t \leftarrow updateRehearsalBuffer(R_t, P_t \cup B_t)$;
$P'_t \leftarrow \emptyset$;
return $\{h'_t, S'_t, P'_t, R'_t\}$;

The updating of the Rehearsal buffer is performed using the state-of-the-art algorithm proposed by [5]. It is a simplified variant of the *prioritized example selection* algorithm [14] that is based on herding. Its main difference, which is in-line with our computational complexity requirements, is that it maintains a running average estimator for each class, in all occasions the buffer is constant and always contain $q = |R|/n$ exemplars per class.

Algorithm 2: RehearsalTraining

Input: model h, state S (global variables), Postponed buffer P, Rehearsal buffer R
Output: updated model h' and state S'

$iter \leftarrow decideIterations(S, P, R)$;
$lrs \leftarrow decideLearningRateSchedule(S, P, R)$;
initialize optimizer with lrs;
forall $i \in 1, \ldots, iter$ **do**
  $D \leftarrow createMixedMiniBatches(i, S, P, R)$;
  foreach *mini-batch* $b \in D$ **do**
    perform one optimizer (e.g. SGD) step with mini-batch b;
return *updated model* h' *and state* S';

4 Alternative Rehearsal Strategies

This section presents a number of alternative ways to perform the rehearsal during training.

4.1 Continuous rehearsal

The baseline strategy, referred as *continuous rehearsal* in the following, assumes that training takes place in each time-step. In each iteration, the latest batch B_t is combined with a different batch $r_j \in R_t$ in order to produce two new batches containing samples from B_t and r_j in a 50-50 ratio. A different batch of samples, r_j, is selected from R_t at each time step t in a round-robin fashion.

A simple experience replay method [2] can be achieved by setting the number of iterations to one. As shown in Section 5, this proves quite effective w.r.t. forgetting, but does not learn new tasks fast enough. This can be solved by increasing the number of iterations at each time-step at the cost of more training times. In what follows, we use `CONR-n` to denote this baseline method with n iterations (`CONR-1` for a single iteration).

4.2 Drift activated rehearsal

In the task-agnostic scenario, where the boundaries of tasks are unknown, we rely on a *concept drift detector* [18] in order to decide *when* to train. The

ECDD detector [16] that we employ uses exponentially weighted moving average charts (EWMA) as an indicator of divergence between samples. This single pass method, with $\mathcal{O}(1)$ update in each time-step, is suitable for performance-critical streaming applications.

$\hat{\mu}_t$ and variance of the misclassification error U_t. It also maintains a more slowly updated running average Z_t, of the error. The following rules are used to decide whether to train:

(i) $Z_t > \hat{\mu}_t + L_t \hat{\sigma}_{Z_t}$, i.e., the running average error estimate must not exceed L_t times the standard deviations above the mean, where L_t is a dynamically updated control limit computed by ECDD.
(ii) $U_t > \hat{\mu}_t + 2\hat{\sigma}_{U_t}$, i.e., the current batch error U_t must not be too high (2 standard deviations above the estimated mean).
(iii) $Z_t > E$, the running average of the error must not exceed a threshold E.
(iv) no training during the last ρ time-steps.

Reasonable choices are $E = 0.2$ for the error threshold, and $\rho = 20$ for the no-training time-steps threshold. Assuming a constant number of iterations for training, the main difference with the `CONR-n` method is that when triggered, the Postponed buffer P will likely contain multiple batches. In each training iteration we iterate over all batches of P. For each batch $p_i \in P$ we read a batch $r_j \in R$ and create two batches which contain 50% from each. Again the buffer R is used in a round-robin fashion using a position pointer that is updated at each time-step. We use `DRIFTA-n` to denote the method which performs a constant number of n iterations when the drift detector is triggered. A variation of this method is to use two detectors, one for the stream B_t as above, and another for the Rehearsal buffer R, by sampling from the Rehearsal buffer uniformly at random. Training takes place when any of the two detectors is triggered, based on the above mentioned heuristics. Let `2DRIFTA-n` denote this strategy with a constant number of n iterations for training.

4.3 Dynamic Number of Training Iterations

So far we have assumed a fixed number of n training iterations per time-step. A more adaptive strategy is to dynamically compute the number of training iterations based on the rate of wrong predictions in the stream. We denote as `DRIFTA-DYN-n` the strategy which relates the estimator Z_t values to the number of iterations. We have experimented with a simple rule which computes the number of iterations as $\lceil 2 * n * log_2(1 + Z_t) \rceil$ where n is a user chosen constant. Similarly, `2DRIFTA-DYN-n` does the same using both drift-detectors by using the maximum of Z_t and $\hat{Z}_t$.

4.4 Iterate until convergence

A different approach in deciding the number of training iterations is to rely on the convergence of the model. In this approach we monitor the loss function $\mathcal{L}$

during the iterations and keep two exponential moving averages, one short and one long. They are updated as

$$\mathcal{A}_s = (1 - \alpha_s)\mathcal{A}_s + \alpha_s\mathcal{L} \tag{1}$$

$$\mathcal{A}_l = (1 - \alpha_l)\mathcal{A}_l + \alpha_l\mathcal{L} \tag{2}$$

with $\alpha_s = 0.5$ and $\alpha_l = 0.05$ respectively. We stop training when the two values converge, i.e. $|\mathcal{A}_l - \mathcal{A}_s| < \epsilon$ for some hyper-parameter value ϵ. This approach can be combined with both continuous and drift-activated approaches resulting in `CONR-CONV`, `DRIFTA-CONV` and `2DRIFTA-CONV`.

4.5 Adjust learning rate

A last action that can affect the efficiency of the rehearsal strategy is to adjust the learning rate across iterations.

The simplest approach is to keep the learning rate η constant to a pre-defined value. Another approach is to use a decay mechanism, e.g. an inverse time decay, that modifies the learning rate through the training iterations. Finally, the drift-activated methods can also use Z_t and a predefined η value to dynamically adjust the initial learning rate. We concluded that a good choice of learning rate can be achieve by using the following function:

$$\mathcal{LR}_{new} = \mathcal{LR}_0 * min(100, 5 * e^{3Z_t}) \tag{3}$$

where LR_{new} is the new initial learning rate and LR_0 is the global, pre-defined learning rate.

5 Experiments

5.1 Experimental setup

Dataset To evaluate the online continual learning strategies of Section we use both MNIST digits [8] and the CIFAR-10 image classification dataset [7].

Following the online continual learning scenario described in Section 3.2, we split the training data into five tasks, where each task contains images from two classes, i.e., $T_i = \{(\mathbf{x}, y)|y \in \{2i, 2i + 1\}\}$, $i = 0, 1, 2, 3, 4$. An online annotated image stream, S, is generated by first sampling multiple images from the first task, then the second task and so on

$$S = (T_0^{(0)}, T_1^{(0)}, T_2^{(0)}, T_3^{(0)}, T_4^{(0)}, T_0^{(1)}, T_1^{(1)}, \cdots) \tag{4}$$

For our experiments, each set $T_i^{(j)}$ has a fixed size of 3200 images, grouped into 100 batches. The model does not know when this change occurs, relying only to the drift detector for feedback. Each batch consists of 32 images sampled randomly from the classes of the current task, T_i. The stream for each experiment has a total length of 1500 batches, with each task appearing three times ($j = 0, 1, 2$). To keep results comparable across experiments, we use the same random seed for sampling, as well as for the initialization of model parameters.

Model and pre-training Following [14] and [5], we use the ResNet32 model for the experiments, and specifically its adaptations to the MNIST and CIFAR-10 dataset, as described by [6]. The MNIST model is trained offline for 15 epochs with 500 images while the CIFAR-10 variant is trained for 100 epochs with a random subset of 15000 images. In both occasions there is an equal number of training images for each of the ten classes. This leads to a model that has gone through a "warm-up" training stage but has not yet been fully trained.

Metrics To evaluate the effectiveness and efficiency of each training strategy, we use the following metrics:

- *Accuracy* (A_t): Accuracy of the model evaluated in the held-out test set, averaged across all tasks.
- *Current task accuracy* (C_t): Accuracy of the model evaluated in the held-out test set, but only for the images belonging to classes of the current task, T_i.
- *Online accuracy* (O_t): Accuracy in each batch of S, evaluated right before it is used for training. This approach was also used in the work of [5].
- *Training iterations* (N_t): Cumulative number of iterations performed during stochastic gradient descent optimization. Given that the model and the batch size is the same across all experiments, this metric that can be used to compare the computational complexity of different training strategies.

Each metric is computed at every step, t, while we report averages across t, e.g., $\bar{A}_t$.

5.2 Experiment 1: The need for continuous rehearsal

This experiment demonstrates the need for an adaptive training strategy in online settings. In figure 2, we compare three training strategies which include (i) no rehearsal (ii) continuous rehearsal with 1 training iteration/experience replay [15] (`CONR-1`), and (iii) 50 training iterations (`CONR-50`). In all cases the learning rate was fixed to $\eta = 0.01$. For the MNIST dataset, it is obvious that `CONR-1` and `CONR-50` are very close in terms of average accuracy, albeit the computational cost of `CONR-50` is 50 times higher. Experiments on the current task accuracy C_t on the CIFAR-10 dataset (Figure 3) pinpoint more accurately the core of the problem for continuous rehearsal strategies. We can see some improvement by `CONR-50`, yet the computational cost is 50 times higher than `CONR-1` which can be prohibitive in resource constrained environments.

5.3 Experiment 2: Choosing the best rehearsal strategy

This experiment evaluates the proposed training strategies. Table 1 contains MNIST results and compares all methods in terms of the training batches (N_t) as well as in terms of the average accuracies $\bar{A}_t$, $\bar{C}_t$ and $\bar{O}_t$. Note that we have added a decay factor for the learning rate (d=0.05) in all methods for a more fair comparison. According to the results for the MNIST dataset, the proposed

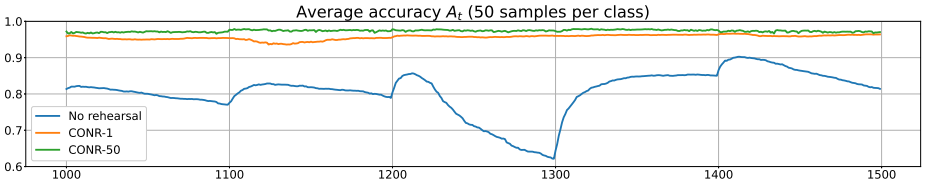

Fig. 2. Results of average accuracy (A_t) for the MNIST dataset. The buffer includes 50 samples per class in the Rehearsal buffer. Continuous rehearsal with one rehearsal `CONR-1` is sufficient. `CONR-50` is slightly better but at the cost of a more resource intensive implementation

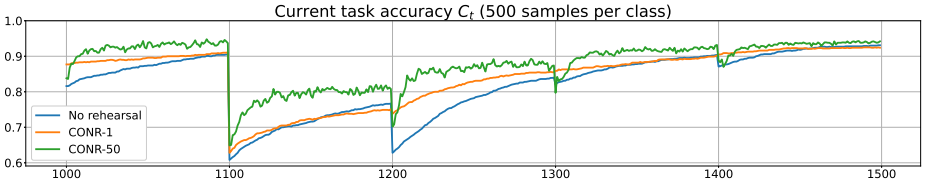

Fig. 3. Results of current task accuracy (C_t) for the CIFAR-10 dataset. the buffer includes 500 samples per class. `CONR-1` performs similar to having no rehearsal at all. `CONR-50` iterations has a clear benefit, but requires 50 times higher computational cost

Table 1. Comparison of the different online training strategies for the MNIST dataset in terms of average values of the metrics across the entire stream. Results are provided for a Rehearsal buffer with 50 samples per class.The highest effectiveness per metric is shown in bold. Numbers in red and blue show the sum of training iterations of the best continuous rehearsal methods and the best proposed methods respectively

Strategy	N_t	$\bar{A}_t$	$\bar{C}_t$	$\bar{O}_t$
`NO REHEARSAL`	1500	0.788	0.93	0.924
`CONR-1`	3000	0.943	0.964	0.963
`CONR-25`	75000	**0.963**	0.984	0.982
`CONR-50`	150000	**0.963**	0.985	0.984
`DRIFTA-DYN-50`	9562	0.961	**0.986**	0.984
`2DRIFTA-DYN-50`	10366	0.962	**0.986**	**0.985**
`DRIFTA-CONV`	3808	0.92	0.983	0.981
`2DRIFTA-CONV`	4982	0.949	0.984	0.982

drift activated dynamic rehearsal methods are most of the times almost equal whilst decreasing training iterations significantly. In CIFAR-10 tests, illustrated in Figure 4, convergence-based methods are even better in terms of computational efficiency with very little compromise in the average metrics. The results clearly demonstrate the benefit of using the proposed drift activated dynamic and convergence rehearsal (especially `2DRIFTA-DYN-n` and `2DRIFTA-CONV`) for the CIFAR-10 dataset. The results show both the limitations of training with

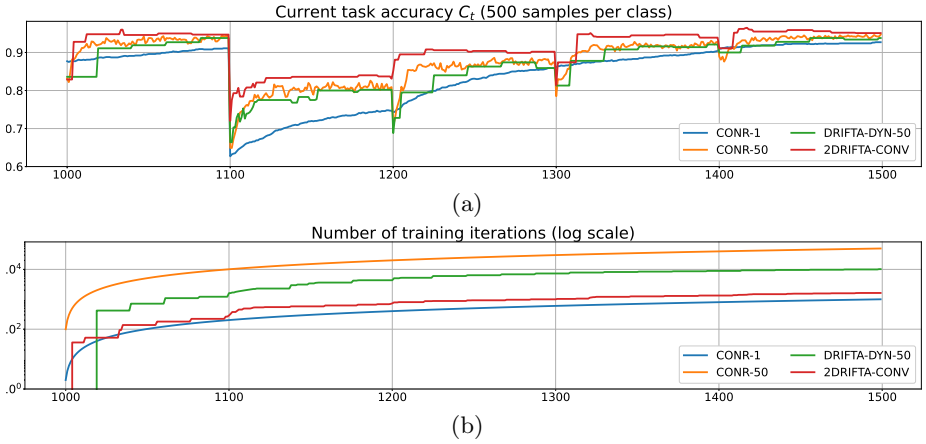

Fig. 4. CIFAR-10 results of (a) Current task accuracy C_t and (b) number of batches N_t for $t = 1000, \cdots, 1500$ of the stream for a Rehearsal buffer with 500 samples per class. `2DRIFTA-CONV` is consistently more effective and significantly more computationally efficient in this experimental setup.

continuous rehearsal methods in an online setting and the benefits of a dynamic approach.

6 Conclusions

This work examined the problem of online continual learning from non i.i.d image streams, with unknown task boundaries, and introduced a generic rehearsal strategy that decides when, as well as how to train. The proposed strategy combines drift detection for the early detection of the task change with methods for determining training parameters (number of training iterations and learning rate) at each training step. The combination of these techniques achieves almost identical current and online task accuracy compared to the static rehearsal strategy baselines, while being more efficient in the use of rehearsal samples, leading to significantly reduced computational cost.

Acknowledgment

This work is supported by the "TEACHING" project that has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No 871385. The work reflects only the author's view and the EU Agency is not responsible for any use that may be made of the information it contains.

References

1. L. Bottou and O. Bousquet, "The tradeoffs of large-scale learning," Optimization for machine learning, p. 351, 2011.
2. A. Chaudhry et al., "Continual Learning with Tiny Episodic Memories," CoRR, vol. abs/1902.10486, 2019.
3. M. Delange et al., "A continual learning survey: Defying forgetting in classification tasks," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.
4. G. Demosthenous and V. Vassiliades, "Continual Learning on the Edge with TensorFlow Lite," arXiv preprint arXiv:2105. 01946, 2021.
5. J. He, R. Mao, Z. Shao, and F. Zhu, "Incremental learning in online scenario," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13926–13935.
6. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv [cs.CV]. 2015.
7. A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.
8. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
9. Z. Li and D. Hoiem, "Learning without Forgetting," CoRR, vol. abs/1606.09282, 2016.
10. A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7765–7773.
11. K. Milan, J. Veness, J. Kirkpatrick, D. Hassabis, A. Koop, and M. Bowling, "The forget-me-not process," in Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 3709–3717.
12. L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent Replay for Real-Time Continual Learning," CoRR, vol. abs/1912.01100, 2019.
13. L. Pellegrini, V. Lomonaco, G. Graffieti, and D. Maltoni, "Continual Learning at the Edge: Real-Time Training on Smartphone Devices," arXiv preprint arXiv:2105. 13127, 2021.
14. S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017, pp. 2001–2010.
15. D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience Replay for Continual Learning," CoRR, vol. abs/1811.11682, 2018.
16. G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," Pattern Recognition Letters, vol. 33, no. 2, pp. 191–198, 2012.
17. F. Wiewel and B. Yang, "Entropy-based Sample Selection for Online Continual Learning," in 28th European Signal Proc. Conf. (EUSIPCO), 2021, pp. 1477–1481.
18. G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," Machine learning, vol. 23, no. 1, pp. 69–101, 1996.