Rehearsal-Free Continual Learning over Small Non-I.I.D. Batches

Vincenzo Lomonaco
University of Bologna
Italy
vincenzo.lomonaco@unibo.it

Davide Maltoni
University of Bologna
Italy
davide.maltoni@unibo.it

Lorenzo Pellegrini
University of Bologna
Italy
l.pellegrini@unibo.it

Abstract

Robotic vision is a field where continual learning can play a significant role. An embodied agent operating in a complex environment subject to frequent and unpredictable changes is required to learn and adapt continuously. In the context of object recognition, for example, a robot should be able to learn (without forgetting) objects of never before seen classes as well as improving its recognition capabilities as new instances of already known classes are discovered. Ideally, continual learning should be triggered by the availability of short videos of single objects and performed on-line on on-board hardware with fine-grained updates. In this paper, we introduce a novel continual learning protocol based on the CORe50 benchmark and propose two rehearsal-free continual learning techniques, CWR* and AR1*, that can learn effectively even in the challenging case of nearly 400 small non-i.i.d. incremental batches. In particular, our experiments show that AR1* can outperform other state-of-the-art rehearsal-free techniques by more than 15% accuracy in some cases, with a very light and constant computational and memory overhead across training batches.

1. Introduction

Consolidating and preserving past memories while being able to learn new concepts and skills is a well-known challenge for both artificial and biological learning systems, generally acknowledged as the plasticity-stability dilemma [23]. In particular, gradient-based architectures are often skewed towards plasticity and prone to catastrophic forgetting when learning over a stream of non-stationary data [4, 32, 12]. A simple solution to deal with this issue would be storing all the data, and cyclically re-train the entire model from scratch [11]. However, this approach is rather impractical when learning from high-dimensional streaming data, especially in highly constrained computational platforms and embedded systems [15, 34].

In recent years, a number of continual learning (CL) strategies have been proposed for deep architectures based

Figure 1: Example images of the 50 objects in CORe50, the continual learning video benchmark used in this paper. Each column denotes one of the 10 categories. Classification experiments in this paper are object-based, so each object corresponds to a class.

on *regularization*, *architectural* or *rehearsal* approaches [24, 17, 2]. Most of the proposals target a Multi-Task (MT) setting where a sequence of independent and tasks are encountered over time. However, for many practical applications, such as natural object recognition, a Single-Incremental-Task (SIT) setting may appear more appropriate [22]. In the SIT setting, we can distinguish three different cases, based on the training batches composition:

1. **New Instances (NI)**: new training patterns of the same classes become available in subsequent batches with new poses and environment conditions (illumination, background, occlusions, etc.).

2. **New Classes (NC)**: new training patterns belonging to different, previously unseen, classes become available in subsequent batches. This is also known as class-incremental learning.

3. **New Instances and Classes (NIC)**: new training patterns belonging to both known and new classes become available in subsequent training batches. To the best of our knowledge, almost no study explicitly addresses the NIC scenario, which we deem as the most natural setting for many applications such as robotics vision, where: i) a large number of small non-i.i.d. training

batches are encountered over time; ii) training batches may contain objects already seen before as well as completely new objects.

Although some researchers pointed out that reducing the size of training batches makes continual learning more challenging [22, 5, 30], we still do not know what is the lower bound for the size of training batches and if it is actually feasible to train a system by gradient descent with very small non-i.i.d. incremental batches each containing few images of a single class. It is well known that stochastic gradient descent (SGD) works well with large and i.i.d. mini-batches, but this assumption is difficult to meet. Let us consider a robot that is learning to recognize some objects shown by an operator (one at the time). In an ideal application, when a new object is shown, the robot acquires a short video and immediately updates its knowledge to become able to recognize the new object. The frames extracted from the video would constitute one or more small mini-batches containing highly correlated patterns from a single class: a rather challenging setting to face with standard SGD-based optimization techniques.

Some rehearsal-based techniques have been proposed in order to mitigate this problem: by maintaining some representative patterns from past experiences, new frames can be interleaved with past ones in each mini-batch. However, this involves extra memory (to store the past data) and computation (due to an higher number of forward/backward steps): in this work we ask ourselves weather continual learning over small non-i.i.d. batches *is feasible without rehearsal*.

The contributions of this paper can be summarized as follows:

- we propose two rehearsal-free continual learning strategies, CWR* and AR1*, as extensions of the CWR+ and AR1 strategies originally proposed for the NC scenario in [22], making them agnostic to the batches composition.

- we show that replacing Batch Normalization with Batch Renormalization [8] allows SGD to continually learn even in the challenging case of very small and non i.i.d. batches.

- we introduce two different approaches, namely *depth-wise layer freezing* and *weight constraining by learning rate modulation* aimed at reducing storage/computation of existing continual learning techniques without hindering accuracy.

- we design and openly release at https://vlomonaco.github.io/core50 a new NIC protocol based on CORe50 [18] to explicitly address non-i.i.d continual learning scenarios (with 79, 196 and 391 training batches). To the best of our knowledge, this is one of the first attempts to scale continual learning

techniques over hundreds of small training batches with real-world highly correlated images.

- we run several experiments to evaluate the proposed strategies (CWR* and AR1*) and to compare them with two baselines and three state-of-the-art rehearsal-free approaches (such as EWC [14], LWF [16] and DSLDA [6]), also in terms of computation and memory efficiency.

2. Continual Learning Strategies

In [22] it was showed that a simple approach like CWR+, where the fully connected layer is implemented as a double memory, is quite effective to control forgetting in the SIT - NC scenario. However, after the first training batch, CWR+ freezes all the layers except the last one, thus losing the benefit of an incremental adaptation of the underlying representation. AR1 [22] was then proposed to extend CWR+ by enabling end-to-end continual training throughout the entire network; to this purpose the Synaptic Intelligence [36] regularization approach (similar to EWC [14]) is adopted to constrain the change of critical weights. In the following subsections we:

1. adapt CWR+ to the NIC scenario, thus making it able to reload past weights for already known classes and to adapt them with weighted contributions from different batches. As AR1 incorporates CWR+ in its main algorithm, this modification will result in two continual learning strategies hereby denoted as CWR* and AR1* (Section 2.1).

2. show that in a complex scenario with small and non i.i.d. batches, Batch Normalization layers thwart the continual learning process and replacing them with Batch Renormalization [8] can effectively tackle this problem (Section 2.2).

3. propose a selective weight freeze for the CNN models adopting Depth-Wise Separable Convolutions (Section 2.3).

4. reduce the computational and storage complexity of AR1 (and in general of EWC like approaches), by introducing an alternative way to implement weights update starting from the Fisher matrix (Section 2.4).

While 1. is specific to CWR+, 2., 3. and 4. can be applied to several other CL approaches as well.

2.1. From CWR+ to CWR*

CWR+, whose pseudo-code is reported in Algorithm 2 of [22] and in the supplementary materials of this work, maintains two sets of weights for the output classification layer: cw are the consolidated weights used for inference and tw the temporary weights used for training; cw are initialized

to 0 before the first batch and then iteratively updated, while tw are reset to 0 before each training batch.

In Algorithm 1, we propose an extension of CWR+ called CWR* which works both under NC and NIC update type; in particular, under NIC the coming batches include patterns of both new and already encountered classes. For already known classes, instead of resetting weights to 0, we reload the consolidated weights (see line 7). Furthermore, in the consolidation step (line 13) a weighted sum is now used: the first term represents the weight of the past and the second term is the contribution from the current training batch. The weight $wpast_j$ used for the first term is proportional to the ratio $\frac{past_j}{cur_j}$, where $past_j$ is the total number of patterns of class j encountered in past batches whereas cur_j is their count in the current batch. In case of a large number of small non-i.i.d. training batches, the weight for the most recent batches may be too low thus hindering the learning process. In order to avoid this, a square root is used in order to smooth the final value of $wpast_j$.

Algorithm 1 CWR* pseudocode: $\bar{\Theta}$ are the class-shared parameters of the representation layers; the notation $cw[j]$ / $tw[j]$ is used to denote the groups of consolidated / temporary weights corresponding to class j. Note that this version continues to work under NC, which is seen here a special case of NIC; in fact, since in NC the classes in the current batch were never encountered before, the step at line 7 loads 0 values for classes in B_i because cw_j were initialized to 0 and in the consolidation step (line 13) $wpast_j$ values are always 0.

1: **procedure** CWR*
2:     $cw = 0$
3:     $past = 0$
4:     init $\bar{\Theta}$ random or from pre-trained model (e.g. on ImageNet)
5:     **for each** training batch B_i:
6:         expand output layer with neurons for the new classes in B_i never seen before
7:         $tw[j] = \begin{cases} cw[j], & \text{if class } j \text{ in } B_i \\ 0, & \text{otherwise} \end{cases}$
8:         train the model with SGD on the s_i classes of B_i:
9:         **if** $B_i = B_1$ learn both $\bar{\Theta}$ and tw
10:         **else** learn tw while keeping $\bar{\Theta}$ fixed
11:         **for each** class j in B_i:
12:             $wpast_j = \sqrt{\frac{past_j}{cur_j}}$, where cur_j is the number of patterns of class j in B_i
13:             $cw[j] = \frac{cw[j] \cdot wpast_j + (tw[j] - avg(tw))}{wpast_j + 1}$
14:             $past_j = past_j + cur_j$
15:         test the model by using $\bar{\Theta}$ and cw

2.2. Replacing Batch Normalization with Batch Renormalization

Batch Normalization (BN) [9] is widely used in modern deep neural networks to control internal covariate shift thus making learning faster and more robust. In BN the mini-batch moments (i.e., mean μ_{mb} and variance σ_{mb}^2) are used

to normalize the input values x_i as:

$$\hat{x}_i = \frac{x_i - \mu_{mb}}{\sqrt{\sigma_{mb}^2 + \epsilon}} \tag{1}$$

where ϵ is a small constant added for numerical stability, and the normalization is per-channel. However, if mini-batches are small and/or non i.i.d. the mini-batch moments are not stable and BN can fail. A natural solution to reduce the moment fluctuations would be replacing μ_{mb}, σ_{mb}^2 with global values μ, σ computed as moving averages over an initial (large-enough) training batch. After all, this is the standard approach when switching from training to inference. However, as argued in [9], using moving averages to perform the normalization during training does not produce the desired effects since gradient optimization and the normalization counteract each other, possibly leading the model to diverge.

Batch Renormalization (BRN) was proposed in [8] to deal with small and non i.i.d. mini-batches. In BRN the normalization takes place as follows:

$$\hat{x}_i = \frac{x_i - \mu_{mb}}{\sigma_{mb}} \cdot r + d \tag{2}$$

$$r = \frac{\sigma_{mb}}{\sigma}, \quad d = \frac{\mu_{mb} - \mu}{\sigma} \tag{3}$$

where μ, σ are computed as moving averages during training. By expanding r and d in the equation 2, we obtain $\hat{x}_i = \frac{x_i - \mu}{\sigma}$ which clearly highlights the dependency on the global moments. A further step is suggested in [8] to clip r in $\left[\frac{1}{r_{max}}, r_{max}\right]$ and d in $[-d_{max}, d_{max}]$. It is worth noting that when $r = 1$ and $d = 0$, then BRN≡BN; hence, by properly setting r_{max} and d_{max} the behavior of BRN can be moved from a pure BN to a more stable normalization based on global statistics. In practice, the author of [8] recommend to perform an initial stage by keeping $r_{max} = 1$, $d_{max} = 0$ in order to stabilize the moving averages μ, σ and then progressively increasing r_{max} and d_{max} to 3 and 5, respectively.

Continual learning over small batches is an emblematic case of small and non i.i.d. minibatches. For example, in NICv2-391 (introduced in Section 3) each training batch includes 300 patterns from a single class, and even using a mini-batch size of 300 (the full batch) patterns remain strongly correlated. Our first attempts to learn continuously over a so long sequence of one-class batches were totally unsatisfactory. Even for the most accurate strategies (e.g., AR1*) accuracy slightly increased in the first batches from 13% to 15% but then remained steady and lower than 16-17%. We initially though that the reason were the single-class mini-batches, making the problem a sort of one-class classification with no negative examples. However, upon replacement of BN with BRN and a proper parametrization, we were able to continuously learn over small batches with
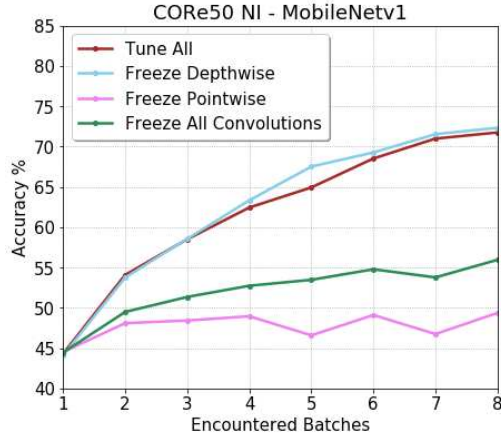
Figure 2: Continual Learning on CORe50, SIT – NI scenario. In the NI scenario all the 50 classes are discovered in the first batch and successive training batches provide new instances of the already known classes; however, since past instances are not retained, the incremental process is prone to forgetting. In this experiment a MobileNet (pre-trained on ImageNet) is incrementally tuned (naïve strategy) over 8 batches with different weight freeze strategies. Each curve is averaged over 10 runs where the batch order is randomly permuted.

unexpected efficacy (see Section 4 for optimal parametrization and results).

2.3. Depthwise Layer Freezing

Depth-Wise Separable Convolutions (DWSC) are quite popular nowadays in many successful CNN architectures such as MobileNet [7, 33], Xception [3], EfficientNet [35]. Classical filters in CNN are shaped as 3D volumes. For example, a $5\times5\times32$ filter spans a spatial neighborhood of 5×5 along 32 feature maps; on the contrary, in DWSC we first perform 32 $5\times5\times1$ spatial convolutions (an independent convolution on each feature maps) and then combine results with a $1\times1\times32$ filter working as a feature map pooler. Advantages in terms of computation and weight reduction have been pointed out by several researchers.

Inspired by previous finding with Hierarchical Temporal Memories [31, 21] where gradient descent by HSR only affects coincidence pooling, here we propose to fine-tune DWSC architectures by freezing depthwise spatial filters and leaving pointwise poolers free to learn. We speculate that modifying a spatial filter (i.e. the way a local neighborhood is processed) can be detrimental in terms of forgetting during continual learning, because it alters the semantics of what upper layers have already learned; on the other hand, feature map pooling, which can be seen as a way to promote feature invariance, is less prone to concept drifts.

A simple experiment is illustrated in figure 2, where a MobileNet is incrementally fine-tuned along the 8 learning

batches of CORe50, SIT – NI scenario [18]. Here, no specific measure is put in place to control forgetting except early stopping the gradient descent after 1 epoch (naïve strategy). The four curves denote the classification accuracy when: i) all the weights are tuned; ii) weights of depthwise convolution layers are frozen; iii) weights of pointwise convolution layers are frozen; iv) weights of all convolution layers are frozen. Note that weights of fully connected layers (e.g. output layer) are never frozen. The proposed strategy (case ii) achieves the best result and, with respect to a full tuning, allows skipping some gradient computations and can reduce the amount of memory used to store weight associated data[1]. The complementary strategy (case iii) is the worst one, thus confirming that altering spatial filters has a strong impact in terms of forgetting.

2.4. Weight Constraining by Learning Rate Modulation

The Elastic Weight Consolidation (EWC) approach [14] tries to control forgetting by selectively constraining the model weights which are deemed to be important for the previous tasks. To this purpose, in a classification approach, a regularization term is added to the conventional cross-entropy loss, where the weights θ_k of the model are pulled back to their optimal value θ_k^* with a strength F_k proportional to their importance for the past:

$$L = L_{cross}(\cdot) + \frac{\lambda}{2} \cdot \sum_k F_k \cdot (\theta_k - \theta_k^*)^2. \qquad (4)$$

Synaptic Intelligence (SI) [36] is a lightweight variant of EWC where, instead of updating the Fisher matrix F at the end of each batch[2], F_k are obtained by integrating the loss over the weight trajectories exploiting information already available during gradient descent. For both approaches, the weight update rule corresponding to equation 4 is:

$$\theta_k^{'} = \theta_k - \eta \cdot \frac{\partial L_{cross}(\cdot)}{\partial \theta_k} - \eta \cdot F_k \cdot (\theta_k - \theta_k^*) \qquad (5)$$

where η is the learning rate. This equation has two drawbacks. Firstly, the value of λ must be carefully calibrated: in fact, if its value is too high the optimal value of some parameters could be overshoot, leading to divergence (see discussion in Section 2 of [22]). Secondly, two copies of all model weights must be maintained to store both θ_k and θ_k^*, leading to double memory consumption for each weight.

---

[1]In per-weight adaptive learning rate methods (such as Adam [13]) extra values (i.e. running averages) need to be stored for each "free" weight. Further, if a regularization method based on Fisher matrix is used (such as EWC [14]) we need to store the optimal value for previous tasks and the Fisher value for each weight.

[2]In this paper, for the EWC and AR1 implementations we use a single Fisher matrix updated over time, following the approach described in [22].
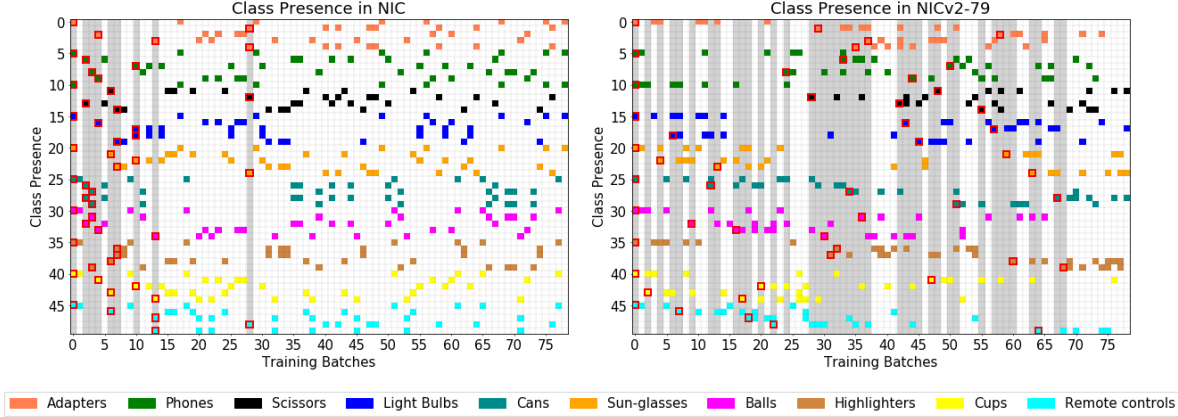
Figure 3: Classes encountered over time in the first run of NIC (left) and NICv2-79 (right). Each row denotes a class; colors are used to group the 50 classes in the 10 categories. Each column denotes a training batch. A colored block in a (row, column) cell is used to indicate that at least one training session of the row class is present in the column batch. Each row contains a maximum of 8 colored blocks, because each class has 8 training sessions in the training set (the remaining 3 sessions are left out for the test set). The red-framed cells denote the first introduction of a class. Gray vertical bands highlight batches where at least one class is seen for the first time. Class presentation ordering for the first run of NICv2-196 and NICv2-391 is also reported in the additional materials of [19].

To overcome the above problems, we propose to replace the update rule of equation 5 with:

$$\theta_k' = \theta_k - \eta \cdot (1 - \frac{F_k}{max_F}) \cdot \frac{\partial L_{cross}(\cdot)}{\partial \theta_k} \qquad (6)$$

where max_F is the maximum value for weight importance (we clip to max_F the F_k values larger than max_F). Basically, the learning rate is reduced to 0 (i.e., complete freezing) for weights of highest importance ($F_k = max_F$) and maintained to η for weights whose $F_k = 0$. It is worth noting that these two updated rules work differently: the former still moves weights with high F_k in direction opposite to the gradient and then makes a step in direction of the past (optimal) values; the latter tends to completely freeze weights with high F_k. However, in our experiments with AR1 the two approaches lead to similar results, and therefore the second one is preferable since it solves the aforementioned drawbacks.

3. CORe50 NICv2

CORe50 [18] was specifically designed as an object recognition video benchmark for continual learning. It consists of 164,866 128×128 images of 50 domestic objects belonging to 10 categories (see Figure 1); for each object the dataset includes 11 video sessions (∼300 frames recorded with a Kinect 2 at 20 fps) characterized by relevant variations in terms of lighting, background, pose and occlusions. The egocentric vision of hand-held objects allows emulating a scenario where a robot has to incrementally learn to recognize objects while manipulating them. Objects are presented to the robot by a human operator who can also provide the labels, thus enabling a supervised classification (such an applicative scenario is well described in [28, 26, 27]).

A NIC protocol was initially introduced for CORe50 [18] where the first training batch contains 10 classes (∼3,000 images) and each of the subsequent 78 incremental batches includes about 1,500 images of 5 classes. However, as shown in Figure 3 (left), the random generation procedure used in [18] produced a sequence where almost all the classes are introduced in the first 10-15 batches making this protocol very close to an NI scenario.

To make the benchmark more challenging and closer to a real application where new objects can be discovered also later in time, we propose a new three-way protocol (denoted as NICv2) where classes first introduction is more balanced over the training batches (see Figure 3, right) and the batch size is progressively reduced, leading to a higher number of fine-grained updates (see Table 1). In particular, in NICv2-391 each of the 390 incremental batches includes only one training session (∼300 images) of a single class. The pseudo-code used to generate the NICv2 protocol is reported in Algorithm 3 of the additional materials of [19].

The test set used for NICv2 is the default test set shared by all the CORe50 protocols [18]; it includes 3 sessions for each class, with null intersection with training batches. Actually, in order to speed up the large number of evaluations (which requires one evaluation after each training batch, repeated for 10 runs) we sub-sampled the test set by selecting 1 frame every second (from the original 20 fps). Because of the high correlation among successive frames in the sequences, such a strong sub-sampling is not reducing the test set variability and the accuracy results on the original and the down sampled version are very close. We made avail-

Table 1: Batch number and composition in NIC and NICv2.

Protocol	# batches	Initial batch		Incremental Batches	
		# Classes	# Images	# Classes	# Images
NIC	79	10	3,000	5	1,500
NICv2-79	79	10	3,000	5	1,500
NICv2-196	196	10	3,000	2	600
NICv2-391	391	10	3,000	1	300

able at all the file lists of the new NICv2 protocols along with the down-sampled test set.

4. Experimental Results

We run several experiments on CORe50 NICv2, to validate the approaches introduced in Section 2 and to compare them with a naïve baseline and three state-of-the-art rehearsal-free approaches. In particular, for all the experiments, the following techniques have been considered:

- CWR*: the extension of CWR+ introduced in Section 2.

- AR1*: the approach introduced in [22], here implemented by replacing CWR+ with CWR* and by adopting the weight constraining by learning rate modulation introduced in Section 2.4.

- Naïve: a baseline technique where we simply continue gradient descent along the training batches and the only measure to control forgetting is early stopping.

- EWC and LWF: the techniques originally introduced in [14] and [16] and adapted to continual learning in SIT scenario as detailed in [22].

- DSLDA: the strategy recently proposed in [6], where an on-line extension of the Linear Discriminant Analysis (LDA) classifier [29] is trained on the top of a fixed deep learning feature extractor. DSLDA obtained state-of-the art accuracy on CORe50 (10 categories setting) [6], even outperforming rehearsal based techniques such as ICARL [30] and ExStream [5].

- Cumulative: this is a sort of upper bound in terms of accuracy because the model is trained on the union of the current batch and all the past data.

For all the experiments we used a MobileNet v1 [7] with: $width\ multiplier = 1$, $resolution\ multiplier = 0.5$ (input 128×128), pre-trained on ImageNet. MobileNet architectures provide a good tradeoff in terms of accuracy/efficiency and, in our opinion, are well suited for porting continual learning at the edge.

For all the above techniques the MobileNet v1 architecture was modified by replacing the 27 Batch Normalization layers with corresponding Batch Renormalization layers and using (for training) a mini-batch size of 128 patterns. We used Batch Renormalization implementation for Caffe [10] made available in [1]. This modification improves accuracy of all the methods, making CWR* and AR1* able to learn also in the case of 391 single class batches. Batch Renormalization hyperparameters and their schedule have been experimentally set as follows:

- **Batch 1**: for the first 48 iterations we keep $r_{max} = 1$, $d_{max} = 0$ to startup the global moments; then, we progressively move r_{max} to 3 and d_{max} to 5 (as suggested in [8]). The weight of the past when updating the moving averages was set to 0.99 (as suggested for $(1 - \alpha)$ in [8]).

- **Subsequent batches**: global moments computed on batch 1 are inherited by batch 2 and slowly updated across the batch sequence. In this case we noted that continual learning over small non i.i.d. batches benefits from more stable moments, and therefore the weight of the past for updating moving averages was set to 0.9999. Here we have no startup phase for the global moment so the values of r_{max} and d_{max} are kept fixed across all the iterations of the batches. While using the suggested values of $r_{max} = 3$ and $d_{max} = 5$ still works, we noted that reducing them (i.e. relaxing batch renormalization constrains) brings some befits. More details about the experiments and the hyperparameters used are provided in the additional materials of [19].

For all the techniques we also applied depthwise layer freezing (as introduced in Section 2.3) starting from Batch 2. This can be simply implemented by setting learning rate to 0 for the 14 non pointwise convolution layers (13 depthwise + 1 3D) in MobileNet v1 architecture. While in NICv2 experiments this had a negligible impact on the accuracy, we found it can be advantageous in other scenarios (see NI curves in Figure 2) and, in general, this reduces computations/storage during SGD (less gradient calculations, lower memory to accumulate per weight extra-data, etc.).

Figure 4 shows the results of our experiments on NICv2-79, NICv2-196 and NICv2-391. The curves are averaged over 10 runs where the training batch order is randomly shuffled. Hyperparameters of the methods have been coarsely
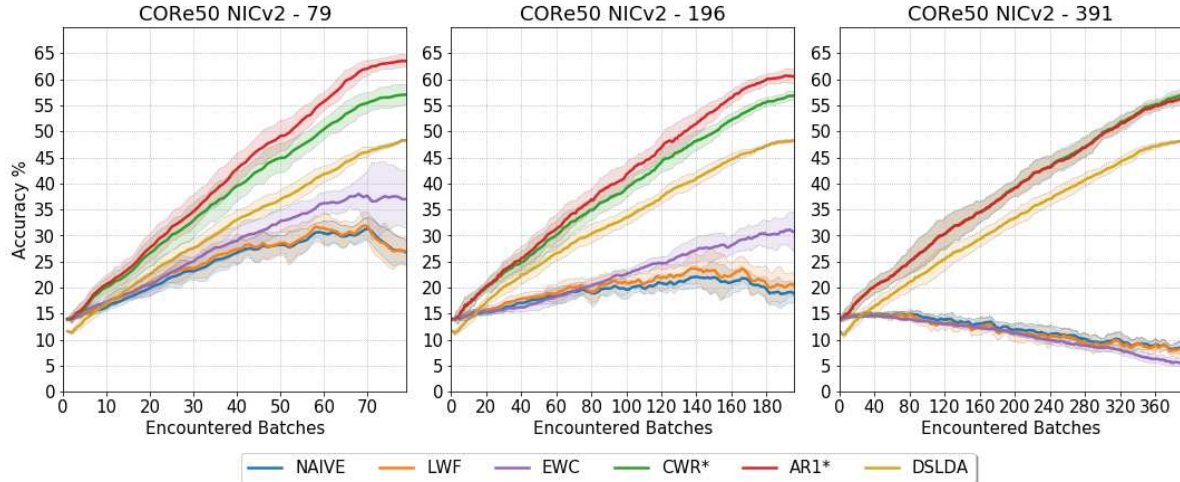
Figure 4: Continual learning accuracy over NICv2-79, NICv2-196 and NICv2-391. Each experiment was averaged on 10 runs. Colored areas represent the standard deviation of each curve. Accuracy performance for the cumulative upper bound (trained on the union of all batches), not reported for visual convenience, is ~85%. Results in tabular form are made available for download at `https://vlomonaco.github.io/core50`.

tuned (i.e., without any systematic grid search) on run 0 and then kept fixed for the other 9 runs. It can be noted that CWR* and AR1* show a very good learning trend across training batches, with only a minor drop in accuracy when the batch granularity decreases. The accuracy near linearly increases for most of the batches and slows down in the final part of the sequences; we believe this is not caused by the saturation of learning capabilities but is more likely due to the absence of example of new classes in the final part of the sequences (see Figure 3b). Standard deviation across runs is also quite small denoting a good stability. Naïve, LWF and EWC exhibit fair performance on 79 batches but their efficacy significantly decreases with 196 batches and are not able to learn in the most challenging case of 391 single-class batches. DSLDA accuracy is quite good and stable but remains lower than CWR* and AR1* in all the three settings. The advantage of AR1* over CWR* (due to the extra freedom to improve the representation) reduces as the batch size decreases and is null for 391 batches. We speculate that, in this case, the gradient steps induced by small and highly non i.i.d. mini-batches tend to overfit the mini-batches themselves with no improvement in terms of generalization.

Figure 5 compares AR1* accuracy in the configuration with Batch Normalization and Batch Renormalization. It is evident that for 391 batches Batch Normalization heavily hurt the learning capabilities. However, it is worth noting that Batch Renormalization brings some advantages to continual learning even when using larger batches that may include patterns from more than one class.

In order to better understand and compare the performance of the proposed continual learning strategies, in Table

Table 2: Total run time (in minutes, for both training and test), memory overhead (in terms of maximum data storage for rehearsal and number of additional trainable parameters introduced) for each strategy on the NICv2-391 protocol. Please note that: *(i)* all the rehearsal-free strategy hereby listed have a *constant* memory / computational overhead which is fixed and independent from the number of training batches processed; *(ii)* the Cumulative metrics are computed considering a re-training from scratch after each incremental batch.

| | | Memory Overhead | |
Strat.	Run Time (m)	Data (MB)	Params (MB)
CWR*	21,4	0	0,2
Naive	25,6	0	0
LWF	27,8	0	0
EWC	31,2	0	24,4
AR1*	39,9	0	12,2
DSLDA	79,1	0	0,2
Cumul.	2826,2	4712,3	0

2 we also report the total run time, the maximum external memory size (where patterns from previous batches are stored) and the number of additional trainable parameters introduced while learning across the NICv2-391 batches. All the metrics are averaged across 10 runs.

Rehearsal-free approaches show a remarkable advantage w.r.t. the cumulative upper bound (where the model is re-trained from scratch after each incremental batch on the cumulated data), both in terms of speed-up and in terms of total memory overhead. Among them, AR1* shows the best trade-off between accuracy and efficiency with about 40 minutes to complete the run and a fixed memory overhead
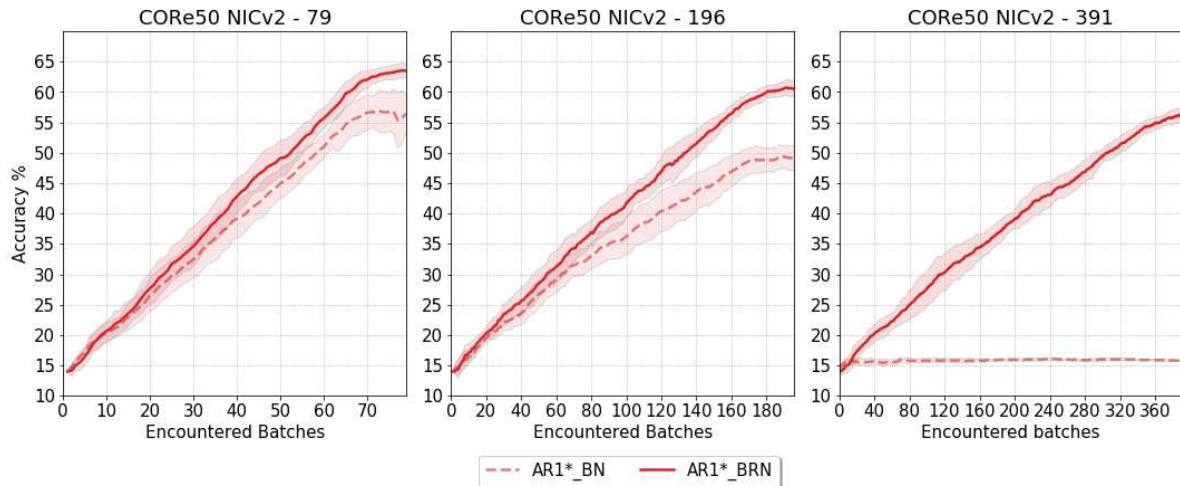
Figure 5: Comparing AR1* accuracy results in the Batch Normalization vs Batch Renormalization configurations.

of only 12.4 MB for handling the additional parameters of the learning rate modulation introduced in Section 2.4. We would also underline that the current Synaptic Intelligence implementation embedded in AR1* is not optimized (gradient is recomputed in python outside the Caffe framework) without exploiting the data already available from SGD. We believe that upon proper optimization, AR1* efficiency can be very close to Naive one.

Finally, it is worth noting that the advantage of weight constraining by learning rate modulation (introduced in Section 2.4) for AR1* is negligible in terms of accuracy (less than 0.1% average improvement in NICv2-79) but relevant in terms of per weight storage since we do not need to store about 3,2 millions θ_k^* values.

5. Conclusions

In this paper, we showed that rehearsal-free continual learning techniques can learn over long sequences of small and highly correlated batches, even in the challenging case of one class at a time. In fact, CWR* and AR1* displayed a good (near linear) learning trend across the training batches and proved to be very robust even with small one-class batches. On the other hand, well known CL techniques such as EWC and LWF were not able to learn effectively in our experiments. We speculate that: (i) a regularization technique alone is not effective to protect important weights in the upper levels when dealing with a large number of small batches; (ii) learning the upper layer(s) "in isolation", as CWR* and AR1* do, is very important for continual learning, especially in SIT setting. DSLDA, that recently achieved state-of-art accuracy on some continual learning benchmarks, performed quite well in our experiments, but its accuracy and efficiency are lower than CWR* and AR1*.

Of course, other continual learning approaches should be

considered in the future for a more comprehensive analysis. For example, here we did not consider rehearsal based approaches such as ICARL [30] and GEM [20] because, even if the use of an external memory to store past data may simplify the task, it brings drawbacks in terms of extra storage/computations. Actually, some preliminary comparisons of CWR+, AR1 and DSLDA with rehearsal-based approaches have been reported in [22] and [6] for CORe50 (NC scenario) showing that the proposed rehearsal-free approaches are still competitive when a moderate number of patterns is maintained in the external memory by ICARL and GEM (2,500-4,500 training images). Another interesting technique, reporting good results on CORe50, is the Dual-Memory Recurrent Self-Organization proposed in [25]: however, results included in that work are not directly comparable with our achievements because the aforementioned approach also exploits the temporal dimension of CORe50 videos (by using temporal windows instead of single frames).

The top accuracy reached by AR1* at the end of the training sequence is in the range 55-65% depending on the batch granularity, and the gap w.r.t. cumulative training ($\sim$85%) exploiting all the data at one time is quite relevant ($>$20%). In the future, we would like to improve the proposed CL techniques to reduce this gap as much as possible. Pseudo-rehearsal, i.e., generating past data without explicit storage, is the main path we intend to explore. Finally, porting continual learning at the edge, i.e. running end-to-end training algorithms on light architectures with neither remote server support nor on-board GPUs, is another topic of interest for us. In the near future, we plan to release a CWR*/AR1* embedded implementation for smart-phones devices and embedded robotics platforms.

References

[1] Seok Hoon Boo. Batch ReNormalization Implementation, 2017. https://github.com/seokhoonboo/caffe-boo. 6

[2] Zhiyuan Chen and Bing Liu. *Lifelong Machine Learning*. Morgan & Claypool Publishers, nov 2018. 1

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:1800–1807, 2017. 4

[4] Robert M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999. 1

[5] Tyler L. Hayes, Nathan D. Cahill, and Christopher Kanan. Memory Efficient Experience Replay for Streaming Learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9769–9776. IEEE, may 2019. 2, 6

[6] Tyler L. Hayes and Christopher Kanan. Lifelong Machine Learning with Deep Streaming Linear Discriminant Analysis. pages 1–15, 2019. 2, 6, 8

[7] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv: 1704.04861*, pages 1–12, 2017. 4, 6

[8] Sergey Ioffe. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1945—-1953, 2017. 2, 3, 6

[9] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning (ICML)*, 10(6):448—-456, aug 2015. 3

[10] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *Proceedings of the ACM International Conference on Multimedia*, pages 675–678, 2014. 6

[11] Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. Fine-tuning deep neural networks in continuous learning scenarios. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10118 LNCS:588–605, 2017. 1

[12] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring Catastrophic Forgetting in Neural Networks. In *AAAI Conference on Artificial Intelligence (AAAI-18)*, New Orleans, USA, 2018. 1

[13] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, pages 1–15, 2014. 4

[14] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neu-

ral networks. In *Proceedings of the National Academy of Sciences*, volume 114, pages 3521–3526, 2017. 2, 4, 6

[15] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual Learning for Robotics. *arXiv preprint arXiv:1907.00182*, pages 1–34, 2019. 1

[16] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *14th European Conference on Computer Vision (ECCV)*, volume 9908 LNCS, pages 614–629, Amsterdam, Netherlands, 2016. 2, 6

[17] Vincenzo Lomonaco. *Continual Learning with Deep Architectures*. Phd thesis, University of Bologna, 2019. 1

[18] Vincenzo Lomonaco and Davide Maltoni. CORe50: a New Dataset and Benchmark for Continuous Object Recognition. In *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, volume 78, pages 17–26, 2017. 2, 4, 5

[19] Vincenzo Lomonaco, Davide Maltoni, and Lorenzo Pellegrini. Fine-Grained Continual Learning. *arXiv preprint arXiv: 1907.03799*, pages 1–14, 2019. 5, 6

[20] David Lopez-paz and Marc'Aurelio Ranzato. Gradient Episodic Memory for Continuum Learning. In *Advances in Neural Information Processing Systems (NIPS 2017)*, 2017. 8

[21] Davide Maltoni and Vincenzo Lomonaco. Semi-supervised Tuning from Temporal Coherence. In *23rd International Conference on Pattern Recognition (ICPR 2016)*, pages 2509–2514, 2016. 4

[22] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73, aug 2019. 1, 2, 4, 6, 8

[23] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4(August):504, 2013. 1

[24] German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, may 2019. 1

[25] German I. Parisi, Jun Tani, Cornelius Weber, and Stefan Wermter. Lifelong Learning of Spatiotemporal Representations With Dual-Memory Recurrent Self-Organization. *Frontiers in Neurorobotics*, 12:1–20, nov 2018. 8

[26] Giulia Pasquale, Carlo Ciliberto, Francesca Odone, Lorenzo Rosasco, and Lorenzo Natale. Teaching iCub to recognize objects using deep Convolutional Neural Networks. In *Proceedings of Workshop on Machine Learning for Interactive Systems*, pages 21–25, 2015. 5

[27] Giulia Pasquale, Carlo Ciliberto, Francesca Odone, Lorenzo Rosasco, and Lorenzo Natale. Are we done with object recognition? The iCub robot's perspective. *Robotics and Autonomous Systems*, 112:260–281, 2019. 5

[28] Giulia Pasquale, Vadim Tikhanoff, Ugo Pattacini, Carlo Ciliberto, Lorenzo Rosasco, and Natale Lorenzo. On the Fly Object Recognition on the R1, Your Personal Humanoid - IIT. https://www.youtube.com/watch?v=HdmDYIL48H4. 5

[29] C. Radhakrishna Rao. The Utilization of Multiple Measurements in Problems of Biological Classification. *Journal of the Royal Statistical Society: Series B (Methodological)*, 10(2):159–193, 1948. 6

[30] Sylvestre Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental Classifier and Representation Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii, 2017. 2, 6, 8

[31] Erik M. Rehn and Davide Maltoni. Incremental Learning by Message Passing in Hierarchical Temporal Memory. *Neural Computation*, 26(8):1763–1809, aug 2014. 4

[32] Anthony Robins. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science*, 7(2):123–146, 1995. 1

[33] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 4

[34] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, and Peter Corke. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, apr 2018. 1

[35] Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019. 4

[36] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 3987–3995, Sydney, Australia, 2017. 2, 4