Supplementary Materials: A Comprehensive Empirical Evaluation on Online Continual Learning

Albin Soutif–Cormerais
Computer Vision Center
Universitat Autònoma de Barcelona
Barcelona, Spain
albin@cvc.uab.cat

Antonio Carta
Department of Computer Science
University of Pisa
Pisa, Italy
antonio.carta@unipi.it

Andrea Cossu
Scuola Normale Superiore
Pisa, Italy
andrea.cossu@sns.it

Julio Hurtado
Department of Computer Science
University of Pisa
Pisa, Italy
julio.hurtado@di.unipi.it

Vincenzo Lomonaco
Department of Computer Science
University of Pisa
Pisa, Italy
vincenzo.lomonaco@unipi.it

Joost van de Weijer
Computer Vision Center
Universitat Autònoma de Barcelona
Barcelona, Spain
joost@cvc.uab.cat

Hamed Hemati
University of St. Gallen
Saint-Gall, Switzerland
hamed.hemati@unisg.ch

A. Appendix

A.1. Additional results

In Table 6 and 7, we present results for more extreme amounts of memory (lower and higher) on Split-Cifar100. For the low memory setting, we notice that the final accuracy results differ more between each method than when using 2000 memory, with ER-ACE getting the best results both in terms of final accuracy and stability. However, the probed accuracy is still close to the one of the i.i.d reference method. When using more memory, we see that the performance of the GDumb baselines matches the one of the i.i.d reference method. However, SCR surpasses both of these, indicating that it's still possible to learn more from the whole stream than from just the memory. We suppose that this is due to its use of a bigger memory batch size, which would be beneficial when a bigger memory size is used. To verify this, we perform an additional experiment where we provide ER with the same memory batch size as SCR, we see that with this modification, the performance of ER matches the one of SCR, indicating that the performance of SCR in this setting is probably due to the bigger memory batch size and not so much to the supervised-contrastive loss.

A.2. Implementation Details

Despite our efforts to make the comparison as fair as possible, there are a few points on which it was hard to make every method coincide. We list them in the following section:

- **Handling of batch normalisation statistics**: While sampling a batch from the current task and the memory, there is a choice that needs to be made when forwarding each batch to the model. The default solution adopted in Avalanche is to concatenate both batches and perform one pass on the model using the concatenated batch statistics (option 1). However, some meth-

[1]Modified ER version with a memory batch size of size 118 (to match the size of the SCR one)

Method	Acc ↑	AAA^{val} ↑	WC-Acc$^{\text{val}}$ ↑	Probed Acc ↑
$i.i.d$	28.3 ± 1.5	-	-	40.0 ± 0.9
GDumb	8.8 ± 0.5	-	-	-
AGEM	3.2 ± 0.4	10.4 ± 0.5	3.2 ± 0.3	19.2 ± 0.7
ER	15.7 ± 1.2	28.6 ± 1.7	7.7 ± 0.9	$\mathbf{38.2} \pm 1.2$
ER + LwF	19.7 ± 1.5	32.5 ± 1.9	10.6 ± 0.9	38.0 ± 1.6
MIR	15.7 ± 1.4	27.4 ± 2.4	9.3 ± 7.7	36.2 ± 1.0
ER-ACE	$\mathbf{20.8} \pm 0.9$	$\mathbf{32.8} \pm 2.2$	$\mathbf{11.5} \pm 0.5$	36.8 ± 1.1
DER++	15.2 ± 1.4	28.9 ± 3.0	7.9 ± 0.6	37.1 ± 1.5
RAR	14.6 ± 1.2	28.6 ± 1.5	7.9 ± 0.6	35.7 ± 0.9
SCR	13.2 ± 0.5	29.4 ± 1.9	8.5 ± 0.5	28.4 ± 0.5

Table 6: Last step results on Split-Cifar100 (20 Tasks) with 500 memory. We report the average and standard deviation over 5 trials

Method	Acc ↑	AAA^{val} ↑	WC-Acc$^{\text{val}}$ ↑	Probed Acc ↑
$i.i.d$	39.0 ± 1.8	-	-	49.3 ± 0.9
GDumb	39.6 ± 0.4	-	-	-
AGEM	3.1 ± 0.3	10.5 ± 0.5	3.1 ± 0.2	18.6 ± 0.8
ER	34.9 ± 1.8	39.1 ± 1.7	13.2 ± 0.8	48.7 ± 0.7
ER + LwF	36.7 ± 1.3	41.7 ± 1.8	17.2 ± 0.9	48.5 ± 0.9
MIR	31.8 ± 1.4	33.6 ± 2.6	8.4 ± 1.4	47.8 ± 0.8
ER-ACE	35.1 ± 1.2	40.6 ± 1.5	16.8 ± 1.1	47.0 ± 0.7
DER++	36.1 ± 1.7	40.8 ± 2.0	14.6 ± 0.4	49.1 ± 0.7
RAR	36.9 ± 2.0	42.2 ± 1.3	16.1 ± 1.2	48.1 ± 1.2
SCR	$\mathbf{43.5} \pm 0.7$	50.2 ± 2.1	$\mathbf{32.6} \pm 0.7$	47.3 ± 0.7
ER[1]	43.0 ± 0.7	$\mathbf{52.7} \pm 2.2$	30.7 ± 0.9	$\mathbf{49.5} \pm 1.4$

Table 7: Last step results on Split-Cifar100 (20 Tasks) with 8000 memory. We report the average and standard deviation over 5 trials

ods were initially implemented by forwarding each batch separately, which could have a huge influence since in that case the separate outputs are created using each internal batch statistics (option 2). In general, while implementing the methods, we chose the option that was working best (ER: 1, DER++: 1, ER-ACE: 2, MIR: 2, SCR: 1, RAR: 2). Note that MIR also updates the batchnorm statistics when forwarding the bigger replay batch (from which it selects the samples to replay), which also has an influence on training that other methods do not have.

- **Memory batch size**: Initially, we wanted to fix the batch size memory using the hyperparameter validation protocol described above, so that each method could select it's adequate memory batch size. However, we found that when using a fixed memory size and doing the hyperparameter selection on only 4 tasks, a big memory batch size was always selected since it was giving more beneficial results after seeing only 4 tasks. This is due to the fact that the optimal use of the full memory size is close to always iterating on samples from the memory. Because of this, we chose to also fix the memory batch size to the same size as the one of the current batch (as done in most works). However, due to its use of a contrastive loss, SCR requires to sample a big batch from the memory, so we fixed the memory batch size to a higher number (118), which makes it behave differently than other methods.

- **Dynamic Classifier**: In continual learning, the learner is not suppose to know the total number of classes it will encounter during the training. This is why we implemented most methods using a dynamic classification layer that adds new units whenever encountering a new class. However, one method (DER++) requires to

replay the logits of samples from previous classes into the new classes. The official implementation made use of a classification layer of fixed size, and we used the same in our experiments, making it different from what other methods do.

A.3. Hyperparameters

In the code (https://github.com/AlbinSou/ocl_survey), we provide the script used to perform the hyperparameter selection (experiments/main_hp_tuning.py) as well as the best configurations we found after 200 trials for each method using the first 4 tasks of the stream. We provide these configurations for each benchmark under the config/best_configs folder. The hyperparameter ranges tested for each method are available in experiments/spaces.py.