An analysis of best-practice strategies for replay and rehearsal in continual learning

Anonymous CVPR submission

Paper ID 41

Abstract

This study is in the context of class-incremental continual learning using replay, which has seen notable progress in recent years, fueled by concepts like conditional, latent or maximally interfered reolay. However, there are many design choices to take when it comes to implementing replay, with potentially very different outcomes in the various class-incremental scenarios. Some of the obvious design choices in generative replay are the use of experience replay (ER), the use of GANs –vs– VAEs, or whether to re-initialize generators after each task. For replay strategies in general, it is an open question how many samples to generate for each new task, and what weights to give generated and new samples in the loss. On top of this, there are many possible CL evaluation protocols differing in the amount of tasks, the balancing of tasks or fundamental complexity (e.g., MNIST -vs- latent CIFAR/SVHN), and thus few generic conclusions about best practices for replay/rehearsal have found consensus in the literature. This study aims at establishing such best-practices by conducting an extensive set of representative replay experiments.

1. Introduction

This article is in the context of class-incremental continual learning (CL), which is considered the most challenging CL scenario among several others, see [47]. Class-incremental CL assumes that data non-stationarities take the form of abrupt switches between mutually exclusive *tasks*, see Fig. 1. One fundamental approach to tackle class-incremental CL is *replay*, which we take to mean the re-use of samples from previous tasks when tackling the current one. In *experience replay*, these samples are taken from a buffer that was populated during previous tasks, whereas in *generative replay*, they are produced by a *generator* trained during previous tasks.

Replay approaches have known considerable success [49] and are actively evolving. Some of the recent additions include latent replay [36], brain-inspired replay[48], maximally interfered replay[1] and adiabatic replay [24].

However, the design space of replay methods is large, which is illustrated in Fig. 2, and it is not clear whether there is a single best-practice strategy that can guide researchers in all possible evaluation scenarios. We can identify several fundamental axes for replay strategies in general, where we omit the issue of using latent replay or not. Rather, we assume that latent replay is used only for problems where it is required.

- Number of samples to replay for each task
- Relative weighting new and generated samples

For generative replay, there are additional choices to make. We believe that the consensus of the community is to use class-conditional generators (see, e.g., [29, 30, 48]) so this is not included here. Similarly, we do not include the choice of a particular form for the involved DNNs, and rather assume that they are chosen according to the characteristics of the data they are applied to.

- Should generators be re-initialized after each task?
- What type of generator should be used, i.e., cVAE or cGAN?

And finally, the chosen evaluation scenario is relevant:

- Number of tasks (small/large)
- Task balancing, i.e., do all tasks contain the same number of classes?
- Fundamental difficulty of the CL problem (e.g., permuted

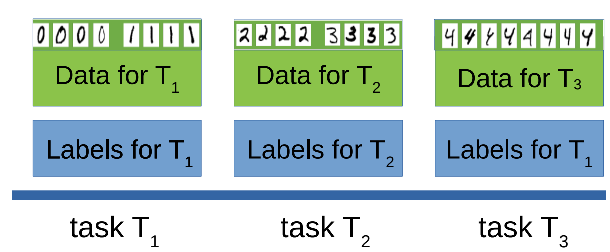


Figure 1. Class-incremental learning, consisting of distinct *tasks* that contain data from pairwise disjoint classes. Please note that not all tasks need contain the same number of classes.

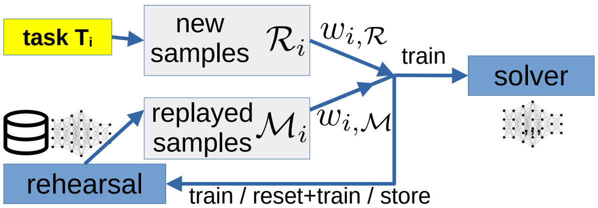


Figure 2. A general depiction of replay approaches, regardless of whether a generator or a buffer is used. For the purposes of this study, we have also shown the different weights $w_{i,\mathcal{R}}$ and $w_{i,\mathcal{M}}$ that real and replayed samples can be assigned in the loss at task i.

MNIST –vs– CIFAR)

The common evaluation scenario seems to be what is usually termed *split-MNIST* and which we denote as D2-2^4, generalized to other 10-class datasets. In D2-2^4 CL problems, the 10 classes are grouped into five tasks of two classes each. Obviously, other tasks can be constructed from 10-class datasets, such as, e.g., D6-1^4 or D1-1^9 which is another common (but less often used) CL benchmark. In any case, most works assume that the number of classes per task is constant and known, which is an assumption that we relax in some of the evaluation of this article.

1.1. Related Work

Many recent works perform comparison studies [11, 32, 48] between different approaches to CL. However, when it comes to rehearsal, no unified view exists w.r.t. various design choices to make. Constant-time rehearsal is used in several studies, combined with weights for replayed and new samples. In some studies [1], weights for replayed samples are chosen by cross-validation, whereas heuristics based on the number of previously seen tasks are used in others [48]. An extensive experimental evaluation of different generator types was performed in [30], with the result that conditional generators are advantageous and that GANs are more suitable than VAEs, although it is not clear how the various parameters were tuned in this study. Although it is rarely indicated in the articles, generators are usually re-initialized after each task, whereas [48] argues for keeping generators since ”preventing forgetting is easier than learning”. To the best of our knowledge, no recent study exists which systematically assesses the performance of rehearsal methods for all of the more common design choices.

1.2. Contributions

This article is the first study to systematically compare different commonly used replay approaches on a wide variety of datasets and dataset splits for continual learning, including the important aspect of latent replay. Based on these investigations, we provide guidelines for using replay-based approaches to continual learning.

2. Methods

2.1. Feature encoding

The training of generative models on complex datasets like SVHN and CIFAR-10 is still challenging [1, 29]. Hence, the use of feature extractors has become a principled approach to deal with this limitation [19, 31, 35, 36, 48]. This study relies upon *supervised contrastive learning (SCL)* [21] based on SimCLR [8] to build a fixed feature extractor to tackle more complex data distributions. Usually, in contrastive learning, the encoding network is trained on large datasets such as ImageNet [42], but our empirical studies have shown that the extracted features might not be beneficial for every scenario, but ultimately depends on the compatibility between the source and target domain. While it might work for e.g., generalizing features from CIFAR-10 and use them for CL training on CIFAR-100 as shown in [48], at the same time they might be insufficient for SVHN and vice versa. We reserve a fixed portion of the original dataset for SCL and exclude these instances from being used for downstream CL, thus the data used for pre-training is identical but not the same. A ResNet-50 with randomly initialized weights is used as the encoding backbone and trained for 256 epochs with a mini-batch size of 256. Each incoming data instance is normalized and augmented by performing a random horizontal flipping and rotation in the range of $-2\% * 2\pi - +2\% * 2\pi$. The final pooling layer outputs a representation vector with the dimensionality $D = (1, 1, 2048)$. The attached projection head consists of two fully-connected layers with 2048 and 128 units respectively using ReLU activation. The n-pairs multi-class loss [44] is used with a temperature of 0.05 and optimized with ADAM using $\epsilon = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. In our experiments, the datasets are transformed prior to CL training, however, an ”on-the-fly” encoding is also feasible at the mini-batch or sub-task level, albeit with poorer runtime efficiency.

2.2. CL strategies

Experience replay (ER) uses reservoir sampling as described in [41], storing 50 samples per encountered class. The time and space complexity is thus constant with respect to the number of classes, even if CL breaks down here at some point when the number of tasks becomes large enough. The ER solver consists of 3 fully-connected layers with 512 units and ReLU activation followed by a softmax output layer.

Deep generative replay (DGR) utilizes cVAEs [22, 45] and GANs [16], whereas the latter is either implemented as a vanilla GAN (cGAN) [33] or by using the Wasserstein distance [3] combined with gradient penalty [17] (WGAN-GP). VAEs use a latent dimension z of 100 and a disentangling factor of $\beta = 1.0$, while GANs use a noise dimension z of 100. WGAN-GP uses a gradient penalty weight of 10 and performs three discriminator iterations per generator iteration. Both, VAEs and GANs are conditioned on the label space by concatenating the output of the label input mapped to a fully connected layer with units either matching the data dimension (cVAE encoder/decoder and cGAN discriminator) or noise dimension (cGAN generator). The VAE encoder consists of two fully-connected layers with 2048 units and ReLU activation, followed by a split output head for the mean vector and logarithmic variance. The decoder is composed of a dense layer chain using ReLU with 128-512-1024 units and a 2048-dimensional output layer with sigmoid activation. The GAN generator is composed of two fully connected layers with 2048 units each and an output layer with sigmoid activation. For cGAN and WGAN-GP each dense layer is followed by a batch normalization layer and LeakyReLU with $\alpha = 0.2$. The Discriminator uses two dense layers with 512 and 256 units followed by LeakyReLU with $\alpha = 0.2$ and Dropout with a rate of 0.3. DGR solvers share the same architecture as the solver used for ER while using the ADAM optimizer with a learning rate of $1\text{e}{-3}$ and $\beta_1 = 0.9, \beta_2 = 0.999$. The learning rate for cVAE encoders and decoders is set to $1\text{e}{-4}$ with $\beta_1 = 0.9, \beta_2 = 0.999$. While cGAN and WGAN-GP use a learning rate of $5\text{e}{-4}$ with $\beta_1 = 0.5, \beta_2 = 0.999$ using the ADAM optimizer. Generators and the ER solver are trained for 100 epochs per task and a mini-batch size of 128. DGR solvers are trained for an additional 50 epochs after generator training. We additionally investigate the effect of re-initializing generators before each new task, as opposed to keeping the same structure (warm-start) for consecutive training.

2.3. Replay strategies

This study focuses on three distinct approaches to replay: balanced, constant and weighted. For the current training task $i > 1$, let $\mathcal{M}_i$ denote the currently replayed samples (either from a buffer or using a generator), $\mathcal{R}_i$ the current (real) task data, and β_{ij} a training mini-batch which is uniformly sampled from $\mathcal{R}_i \cup \mathcal{M}_i$ (see also Fig. 2). Further, we define a parameter $\chi_\mathcal{M}$ that defines the proportion of replayed samples at each task, therefore also defining the proportion of replayed samples in each training mini-batch:

$$\chi_\mathcal{M} = \frac{|\mathcal{M}_i|}{|\mathcal{R}_i| + |\mathcal{M}_i|}, \qquad \chi_\mathcal{R} = 1 - \chi_\mathcal{M}. \tag{1}$$

The **balanced** strategy ensures a linear scaling of $\mathcal{R}_i$ w.r.t. previously encountered classes. Denoting the number of classes for each task j as N_j, we can ensure that amount of samples from each class in all mini-batches β_{ij} is identical by choosing:

$$\chi_\mathcal{M} = \frac{\sum_{j=1}^{i-1} N_j}{\sum_{j=1}^{i} N_j} \tag{2}$$

The **constant** strategy generates an amount of samples identical to the amount of samples in $\mathcal{R}_i$. Here, storage consumption and re-training time is bounded, and $\chi_\mathcal{M}$ is set to 0.5. There are works which replay a constant number of samples regardless of the size of $\mathcal{R}_i$, which however implicitly assumes that all tasks contain the same amount of samples. Please note that classes will generally be unbalanced in each mini-batch for this strategy.

The **weighted** strategy is a direct extension to the constant strategy, which implements an additional mechanism to ensure balancing. This approach is inspired by [48] and uses distinct weights $w_{i,R}$ for real samples and $w_{i,M}$ for generated ones, which are applied to the loss function $\mathcal{L}$ to offset imbalances. Here, the loss is split into two parts: $\mathcal{L}_\mathcal{R}$ and $\mathcal{L}_\mathcal{M}$ computed from real and generated samples, respectively (see also Fig. 2). In its original formulation the calculation of these balancing coefficients is based on the amount of encountered tasks, which we will refer to as *task-weighted loss weights* (see Eq. (3)). We additionally added an adaptation based on class counts, which we term *class-weighted loss weights* (see Eq. (4)). Assuming that the amount of samples per class is roughly similar, we compute these weights according to:

$$\mathcal{L} = w_{i,\mathcal{R}}\mathcal{L}_\mathcal{R} + w_{i,\mathcal{M}}\mathcal{L}_\mathcal{M} = \frac{1}{i}\mathcal{L}_\mathcal{R} + \frac{i-1}{i}\mathcal{L}_\mathcal{M} \tag{3}$$

$$\mathcal{L} = w_{i,\mathcal{R}}\mathcal{L}_\mathcal{R} + w_{i,\mathcal{M}}\mathcal{L}_\mathcal{M} = \frac{1}{\sum_{j=1}^{i} N_j}\mathcal{L}_\mathcal{R} + \frac{N_i}{\sum_{j=1}^{i} N_j}\mathcal{L}_\mathcal{M}. \tag{4}$$

For generative replay, loss weighting is applied to both the generator and solver losses.

3. Experiments

3.1. Evaluation protocol

Usually CIL is investigated in an artificially composed setting where tasks share the same amount of classes per task which results in approximately evenly balanced compositions [2, 30, 40, 43, 50]. We argue that this assumption seems highly unrealistic for real world learning scenarios, since novel additions to a persistent knowledge base should diminish and fluctuate over the course of facing a multitude of separate training sessions. We share the idea that artificial CIL scenarios, despite their usability for prototypical evaluation, should be adapted and expanded [10]. We aim to extend common CL benchmarks towards a more sophisticated CIL evaluation protocol to use for future research in this area. Still, some relaxation has to be accepted in order to enable tractable experimentation, replay is investigated assuming known task-boundaries and disjoint classes, while it is assumed that data from all tasks occurs with equal probability. Data is normalized to a range of $[0, 1]$ and randomly shuffled. We perform a two-staged training, with an initial run on T_1 and a sequence of replay runs $T_i, i > 1$. Further, we allow no information about tasks in advance, except for the knowledge of present classes and the amount of samples per incoming task. This study investigates three directions for task composition to model CIL-problems (CP) Tab. 1. We divide them into: *usual* problems (U-CP), commonly found in CIL literature (e.g. D5-5, D2-2^4, D1-1^9), and the more imbalanced, *diminishing* (D-CP) and *alternating* problems (A-CP). With the latter two showing an distinctive approach to model variance in task composition. D-CP reflects a decline in the amount of novel data over the course of training, while A-CP models a constant change in the number of arriving class additions to the knowledge base. Classes per task are randomly selected once and fixated throughout experimentation.

split ↓	dataset↓ MNIST / F-MNIST SVHN / CIFAR10	E-MNIST
U-CP1	D5-5	/
U-CP2	D2-2^4	/
U-CP3	D1-1^9	/
D-CP1	D4-3-2-1	D20-1^{10}
D-CP2	D5-1^5	/
A-CP1	/	D2-10-3-10-5
A-CP2	/	D10-2-10-3-10

Table 1. This table shows all task compositions evaluated in the empirical study. A composition consists of a sequence of classes, whereas each number indicates the total amount of classes encountered in each separate training phase. D2-2^4 shall be interpreted as 2-2-2-2-2 and thus has a total of 5 tasks with 2 classes each.

3.2. Data

MNIST [26] consists of 60.000 28×28 grayscale images of handwritten digits (0-9).

Fashion-MNIST [51] consists of 60.000 images of clothes in 10 categories and is structured like MNIST.

E-MNIST [9] is an extended version of MNIST and contains additional letters. A total of 131.000 samples are balanced across 47 classes, and thus allows to model a CL problem where the amount of already acquired knowledge can be significantly larger than the amount of new data added with each successive task.

SVHN [34] contains 60.000 RGB images of house numbers (0-9, resolution 32×32). This dataset is imbalanced, as classes 1 and 2 are overrepresented, while classes 0 and 9 are underrepresented.

CIFAR-10 [25] contains 60.000 RGB images of natural objects, resolution 32x32, in 10 balanced classes.

Feature encoding was used for SVHN and CIFAR-10 with a pre-trained feature-extractor to reduce the complexity of the data as discussed in Sec. 2.1. For SVHN, we take half of the extra split, while we divide the CIFAR10 training split in half, reserving one part for pre-training and the other for downstream CL. No encoding was performed for MNIST, FashionMNIST and E-MNIST.

3.3. Evaluation metrics

The accuracy α_{ij} of a solver S_i after each training phase $T_j, 1, \ldots, j$ is evaluated on a corresponding held-out test set. The final accuracy α_{end} is evaluated on a joint test set composed from samples of all present classes, and reported after training on the complete task sequence. For comparison, we also provide the joint-training performance α_{base}, achieved by a default solver on the union of all classes from each distinct dataset. To measure CL capacity we define forgetting F_{ij}, as an averaged value over all tasks F_T which is defined as follows:

$$F_{ij} = \max_{i \in \{1,..,T-1\}} \alpha_{ij} - \alpha_{Tj}, \qquad \forall j < T.$$

$$F_T = \frac{1}{T-1} \sum_{j=1}^{T-1} F_{Tj}, \qquad F_T \in [0, 1]. \tag{5}$$

3.4. Results

The experiments are conducted on a cluster of 25 machines equipped with single RTX3070Ti GPUs. Five randomly initialized runs were performed for all configurations on the task compositions showcased in Tab. 1. We also offer a publicly available TensorFlow2 implementation [1]. The results are presented in the following order: First, a comprehensive comparison of the investigated CL methods from Sec. 2.2 in their unmodified version in a memory-constrained (constant) scenario is presented to investigate the impact of different datasets and task splits (see Sec. 3.1). Next, the impact of applying the proposed replay modifications as described in Sec. 2.3 is assessed and a final evaluation of resetting and reusing generators for DGR is performed.

Evaluation of the memory-constrained scenario for unmodified CL methods are displayed in Tab. 2. We've identified CVAEs to be most effective on MNIST and Fashion-MNIST for almost every investigated task split, while ER performs stronger on encoded features. GAN-based DGR shows the weakest results across all datasets, especially having difficulties with longer task sequences and sequentially learning the latent feature representations. Additionally, conditional GANs regularly suffer from major convergence problems and mode collapse [39, 46], especially on

[1]The code and instructions to reconstruct the experiments can be found under the following link

		method↓ ER	CVAE	CGAN	WGAN-GP
MNIST / F-MNIST	U-CP1	.85 ±.01 / .27	**.97** ±.00 / **.03**	.93 ±.01 / .11	.95 ±.01 / .08
		.71 ±.01 / .47	**.78** ±.01 / **.34**	.64 ±.01 / .61	.69 ±.01 / .52
	U-CP2	.76 ±.01 / **.05**	**.93** ±.00 / .08	.20 ±.01 / 1.0	.88 ±.01 / .14
		.60 ±.04 / .49	**.63** ±.03 / **.44**	.48 ±.01 / .63	.44 ±.01 / .69
	U-CP3	.15 ±.11 / .64	**.83** ±.01 / **.19**	.10 ±.01 / 1.0	.70 ±.06 / .32
		.37 ±.03 / .81	**.67** ±.04 / **.41**	.20 ±.19 / .91	.54 ±.03 / .51
	D-CP1	.86 ±.01 / .10	**.95** ±.01 / **.03**	.10 ±.00 / .49	.93 ±.00 / .05
		.64 ±.01 / .20	**.66** ±.04 / **.18**	.55 ±.03 / .25	.59 ±.00 / .23
	D-CP2	.84 ±.00 / .21	**.91** ±.00 / **.07**	.57 ±.40 / .47	.83 ±.05 / .10
		.69 ±.01 / .36	.65 ±.01 / **.27**	.51 ±.06 / .40	.50 ±.04 / .33
SVHN / CIFAR10	U-CP1	**.81** ±.03 / **.25**	.68 ±.04 / .49	.45 ±.00 / .94	.56 ±.05 / .72
		.62 ±.01 / **.40**	.54 ±.02 / .63	.40 ±.01 / .87	.44 ±.02 / .79
	U-CP2	**.78** ±.03 / **.26**	.54 ±.05 / .53	.15 ±.00 / .98	.35 ±.03 / .78
		.62 ±.02 / **.36**	.44 ±.03 / .53	.19 ±.01 / .93	.39 ±.03 / .79
	U-CP3	**.56** ±.40 / **.39**	.43 ±.05 / .60	.06 ±.00 / 1.0	.30 ±.01 / .72
		.38 ±.25 / **.63**	.34 ±.03 / .71	.10 ±.01 / 1.0	.28 ±.04 / .78
	D-CP1	**.82** ±.00 / **.11**	.56 ±.04 / .28	.09 ±.01 / .48	.41 ±.05 / .41
		.60 ±.02 / **.16**	.42 ±.03 / .27	.10 ±.00 / .43	.36 ±.05 / .34
	D-CP2	**.75** ±.09 / **.26**	.58 ±.01 / .43	.20 ±.00 / .99	.44 ±.01 / .61
		.65 ±.01 / **.32**	.36 ±.04 / .57	.10 ±.00 / .97	.32 ±.02 / .67
E-MNIST	D-CP1	**.64** ±.02 / .47	.44 ±.03 / **.34**	.21 ±.15 / .70	.22 ±.01 / .37
	A-CP1	.54 ±.03 / .52	**.71** ±.02 / **.36**	.17 ±.01 / .96	.59 ±.03 / .47
	A-CP2	**.64** ±.02 / .47	.63 ±.02 / **.45**	.55 ±.03 / .52	.55 ±.03 / .52

MNIST	F-MNIST	SVHN	CIFAR-10	E-MNIST1	E-MNIST2	E-MNIST3
.98	.88	.92	.75	.89	.89	.88

Table 2. Experimental results. **Upper table** Shows the results for all investigated CL methods in their unmodified settings while following the constant replay scenario (see Sec. 3.1). We present the final test-set accuracy α_{end} followed by average forgetting F_{end} for each CIL problem presented in Tab. 1. **Lower table** The joint-training baselines for all datasets. We've used the solver network as described in Sec. 2.2 trained for 100 epochs as the classification model. E-MNIST1/2/3 refer to the joint class sets as apparent in D-CP1, A-CP1 and A-CP2. All results are averaged across $N = 5$ runs.

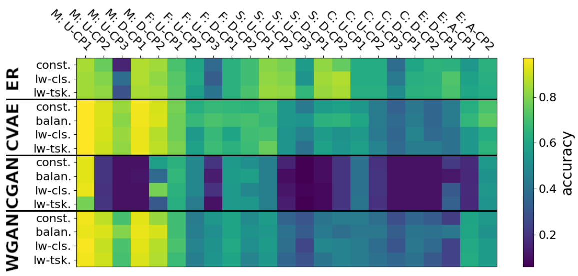
encoded SVHN and CIFAR-10. Although, WGANs with GP show competitive results when directly compared to CVAEs, they come with the major drawback of increased training time. An epoch of generator training on U-CP3 for SVHN takes 25 seconds per epoch for CVAE, 52s/epoch for CGAN and 90s/epoch for WGAN-GP. Regarding the usage of common CIL task splits, we have identified problems like U-CP1 (5-5) as vacuous to evaluate in this context due to the low number of individual replay training sessions and the inherent balance in terms of the set of classes that each task represents. We also observe this to some extent for longer and equally balanced task sequences like U-CP2 (2-2^4) and U-CP3 (1-1^9), as long as the initial capacity of the buffer or generator allows to capture the data somewhat effectively. This is reflected by the small margin in terms of accuracy and forgetting between U-CP2 and U-CP3 despite the latter objective doubling the amount of sequential learning tasks. Additionally, DGR-CVAE for example, reaches a similar performance for D-CP1 (3 tasks and 10 classes in total) as for U-CP3 (10 tasks and 10 classes in total). We also observe that all CL methods struggle to reach satisfactory results on tasks compositions where the amount of new data to learn diminishes steadily over the course of a growing number of learning experiences, as can be seen for E-MNIST D-CP1 (20-1^{10}). We've also gathered more interesting results, like e.g., the poor performance of ER on MNIST/F-MNIST U-CP3. Here, the final accuracy is far off from our expectation, which again shows that minor changes in the evaluation protocol, such as randomized class orders, may show very different results than usually found in the literature.

Results for the application of proposed replay modifications can be found in Fig. 3, which showcases an comprehensive overview over their benefits for CL training. The corresponding values for forgetting can be found in Sec. 6 Fig. 4. For ER, an explicit loss weighting has shown to be beneficial especially considering longer task sequences. However, we mostly couldn't distinguish a significant difference between the resulting performance of weighting on a class basis – versus – weighting on a task basis except for E-MNIST D-CP1 where balancing the loss coefficients based on the class count outperforms the weighting strategy based on the number of tasks, a training on longer task sequences like D20-1^{20} could amplify this effect even more. For DGR-based rehearsal, we identified the **balanced scenario** as the most stable one, achieving the highest accuracy and least forgetting during replay training. While there are definitely some cases where explicit loss-weighting might be on par with a balancing strategy, these rare cases seem to occur very rarely or for task splits where the replay strategy has no significant influence (e.g. U-CP1).

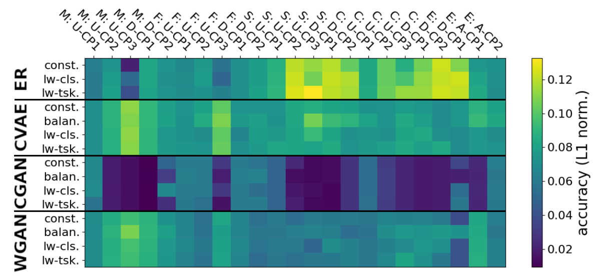
Re-using or re-starting generators does not make much difference, as described by Fig. 3 (e.) and (f.). We found only marginal increases and decreases in accuracy and forgetting across all experiment groups, which may be due to statistical regularities.

4. Discussion

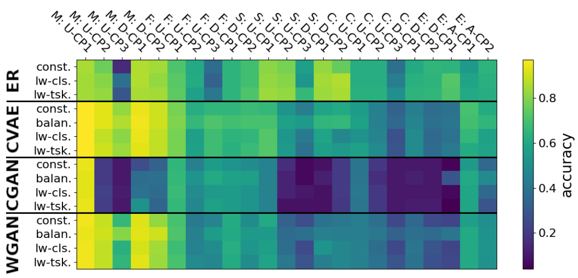
Construction of proper CIL evaluation protocols: The use of simplified benchmarks without the use of full-fledged protocols can mask the weaknesses of replay methods in CL. We have identified some criteria to consider in a CIL scenario. More interesting benchmarks should include long task sequences within a limited computational and memory budget, as discussed in [14]. However, we have empirically confirmed that the length of the task sequence alone is not a clear indicator of the overall complexity of an objective, but rather must be considered as one piece of the puzzle when constructing appropriate CIL benchmarks. We assume that the mixture of the total amount of samples/classes, their



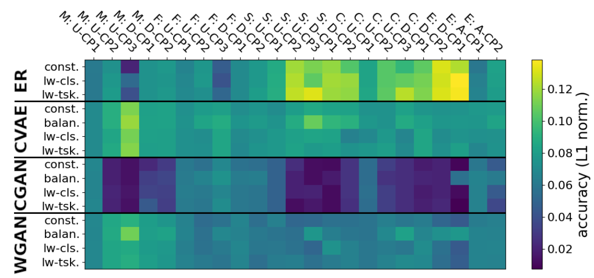
(a) Absolute accuracy (warm-start).



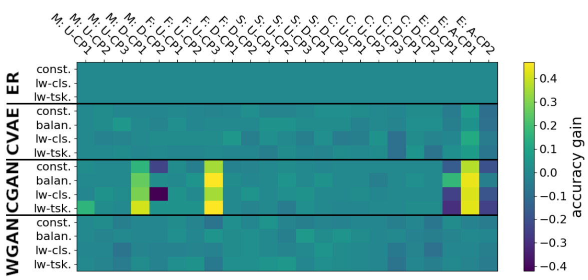
(b) Accuracy normalized over each column vector (warm-start).



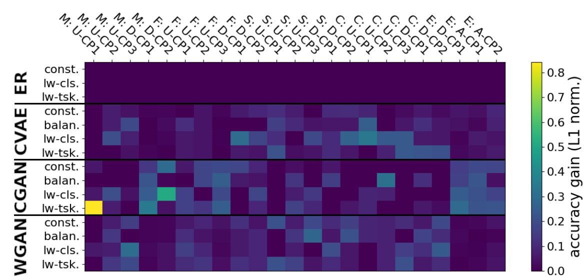
(c) Absolute accuracy (generator reset).



(d) Accuracy normalized over each column vector (generator reset).



(e) Difference in absolute accuracy between (c.) and (a.).



(f) Difference in normalized accuracy between (c.) and (a.).

Figure 3. This illustration showcases the final accuracy α_{end} for all investigated datasets/splits. Each column represents a distinct task split on each dataset, whereas the first letter (”M”, ”F”, ”S”, ”C” stands for MNIST, Fashion-MNIST, SVHN and CIFAR-10), followed by an task split descriptor from Tab. 1. The deployed CL methods (4 groups * N rows) were modified as explained in Sec. 2.3: *const.* = constant, *balan.* = *balanced* (DGR exclusive), *lw-cls.* = loss-weighting by class count, *lw-tsk.* = loss-weighting by task count. The results for ER (top-most three rows) are re-used and serve as a baseline for the results in (c.-f.).

temporal occurrence and balancing in each training phase, as well as the resulting interference between already captured and newly arriving data statistics are of central importance and must be considered holistically. We therefore argue for the introduction of a meaningful CIL evaluation that takes these points into account, rather than relying on some common but often uninformative CIL problems used in the literature just for convenience. What we have not yet constructed is a natural imbalance for the number of samples per class, which would be a nice addition to extend our experiments. Furthermore, an aspect such as natural repetition [10] would be a nice addition to the diminishing and alternating splits to render a CIL problem more realistically. An efficient knowledge adaptation is required here, and the CL method has to be able to deal with repetitive patterns from a previous distribution while encountering new data to add to its knowledge base. The experimental evaluation also showed that the task order plays an important role in the evaluation, see e.g. ER on Fig. 3a M: U-CP3. These results are far from the results of other empirical studies on exactly the same split [4]. This could be due to the fact that the solver's parameter set θ_T at the end of training resides in the same low-loss region of the first task, since the same network is reused and not reinitialized, in contrast to e.g., GDumb [37]. This should underline the need for stronger randomization and its crucial role in creating meaningful

CIL experiments. We also make a more practical reference to a real-time application with splits such as D-CP1, D-CP2 and A-CP1, A-CP2, since a CL model will eventually reflect accumulated knowledge about a larger corpus of previously collected data and therefore needs to be able to adapt to the assumption that new data will only represent a small fraction of the total knowledge.

Identifying efficient generators for replay: Since factors such as memory consumption and a limited compute budget of CL methods are undeniably important metrics [12, 18, 38], we should not be guided solely by the resulting accuracy when evaluating the usability of a method. We find that conditional VAEs are the best-performing generative model in the context of the replay efficiency and resulting solver accuracy for DGR, as CGANs often suffer from mode collapse [5, 39, 46], while WGANs with GP are slower by a factor of three during training. When combining generative models with the presented replay strategies, we identified the balanced scenario as the best performing one. This is in line with the views of [30], where it was observed that to ensure a balanced distribution of classes, the number of generated samples must be rescaled linearly with respect to the number of tasks to ensure stable generators. However, this approach is accompanied by a worse runtime and a higher consumption of intermediate memory to compensate for the loss of knowledge. It would also be interesting to investigate whether there is a perfect timing for replaying certain aspects of the data as discussed in [23], and combine this with a dynamically balanced replay mechanism, as this could also reduce the reliance on sharp task boundaries to trigger generator re-training, which is a serious limitation in any streaming data setup.

The use of latent replay: Pre-trained (PT) models can be advantageous for several replay methods, and these advantages can vary greatly from algorithm to algorithm, while furthermore there appears to be different behavior for different types of PT models used [27]. Generative models are still limited in their capacity to model more complex distributions [2, 28] and therefore rely on PT models to be useful for datasets like SVHN and CIFAR-10/100. Our study uses supervised contrastive learning on the same data domain, but this could also be adapted to the self-supervised learning paradigm to better fit a real CL setting [6, 7, 13]. Empirical studies have already shown that PT models can be combined with CL algorithms and applied to incremental batch learning [15] as well as to learning from streaming data [20].

5. Conclusion and take-home messages

The present study could be extended in several direction, mainly by varying more hyper-parameters, such as number of training epochs or generator/solver structure. Based on the presented results, we can formulate the following take-home messages for CL practitioners:

Balanced replay performs best This is observable across virtually all dataset splits and generative replay methods (WGAN and CVAE), but is most apparent for unbalanced task splits. The other weighting schemes we tested can be competitive for some datasets, but perform generally worse.

Warm-starting is feasible but not required Although warm-starting can considerably improve convergence time, the final accuracies do not seem to depend on this choice at all.

ER is competitive for latent replay Although ER does not show the best performance for simple datasets, it excels when latent data are replayed. This is presumably since latent features are harder to model by generative models, and the replay of real samples gives an advantage.

Use CVAEs as generators Although generators based on WGAN-GP can be competitive (but not superior), they suffer from very long training times and the need to tune the number of training epochs via cross-validation which is in principle inadmissible. For CVAEs, early-stopping can be used since they minimize a loss function, which GANs do not.

References

[1] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval, 2019. 1, 2

[2] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval, 2019. 3, 7

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017. 2

[4] Benedikt Bagus and Alexander Gepperth. An investigation of replay-based approaches for continual learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2021. 6

[5] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Seeing what a gan cannot generate. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4502–4511, 2019. 7

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020. 7

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 7

[8] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020. 2

[9] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017. 4

[10] Andrea Cossu, Gabriele Graffieti, Lorenzo Pellegrini, Davide Maltoni, Davide Bacciu, Antonio Carta, and Vincenzo Lomonaco. Is class-incremental enough for continual learning?, 2021. 3, 6

[11] 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*, 44(7):3366–3385, 2021. 2

[12] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018. 7

[13] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597, 2021. 7

[14] Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning, 2019. 5

[15] Jhair Gallardo, Tyler L Hayes, and Christopher Kanan. Self-supervised training enhances online continual learning. *arXiv preprint arXiv:2103.14010*, 2021. 7

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2

[17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017. 2

[18] Md Yousuf Harun, Jhair Gallardo, Tyler L. Hayes, and Christopher Kanan. How efficient are today's continual learning algorithms?, 2023. 7

[19] Tyler L. Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting, 2020. 2

[20] Dapeng Hu, Shipeng Yan, Qizhengqiu Lu, Lanqing Hong, Hailin Hu, Yifan Zhang, Zhenguo Li, Xinchao Wang, and Jiashi Feng. How well does self-supervised pre-training perform with streaming data? *arXiv preprint arXiv:2104.12081*, 2021. 7

[21] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020. 2

[22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[23] M Klasson, H Kjellström, and C Zhang. Learn the time to learn: Replay scheduling in continual learning. *Transactions on Machine Learning Research*, 9, 2023. 7

[24] Alexander Krawczyk and Alexander Gepperth. Adiabatic replay for continual learning, 2023. 1

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4

[26] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989. 4

[27] Kuan-Ying Lee, Yuanyi Zhong, and Yu-Xiong Wang. Do pre-trained models benefit equally in continual learning?, 2022. 7

[28] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning, 2018. 7

[29] Timothée Lesort, Hugo Caselles-Dupré, Michael Garcia-Ortiz, Andrei Stoian, and David Filliat. Generative models from the perspective of continual learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019. 1, 2

[30] Timothée Lesort, Alexander Gepperth, Andrei Stoian, and David Filliat. Marginal replay vs conditional replay for continual learning. In *International Conference on Artificial Neural Networks*, pages 466–480. Springer, 2019. 1, 2, 3, 7

[31] Zheda Mai, Ruiwen Li, Hyunwoo Kim, and Scott Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning, 2021. 2

[32] Marc Masana, Xialei Liu, Bartlomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *arXiv preprint arXiv:2010.15277*, 2020. 2

[33] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[34] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 4

[35] Oleksiy Ostapenko, Timothee Lesort, Pau Rodríguez, Md Rifat Arefin, Arthur Douillard, Irina Rish, and Laurent Charlin. Continual learning with foundation models: An empirical study of latent replay, 2022. 2

[36] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209. IEEE, 2020. 1, 2

[37] 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*, pages 524–540. Springer, 2020. 6

[38] Ameya Prabhu, Hasan Abed Al Kader Hammoud, Puneet Dokania, Philip H. S. Torr, Ser-Nam Lim, Bernard Ghanem,

and Adel Bibi. Computationally budgeted continual learning: What does matter?, 2023. 7

[39] Eitan Richardson and Yair Weiss. On gans and gmms. *Advances in Neural Information Processing Systems*, 31, 2018. 4, 7

[40] Amanda Rios and Laurent Itti. Closed-loop memory gan for continual learning. *arXiv preprint arXiv:1811.01146*, 2018. 3

[41] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32:350–360, 2019. 2

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 2

[43] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*, 2017. 3

[44] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016. 2

[45] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015. 2

[46] Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *2020 international joint conference on neural networks (ijcnn)*, pages 1–10. IEEE, 2020. 4, 7

[47] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 1

[48] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):1–14, 2020. 1, 2, 3

[49] Eli Verwimp, Matthias De Lange, and Tinne Tuytelaars. Rehearsal revealed: The limits and merits of revisiting samples in continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9385–9394, 2021. 1

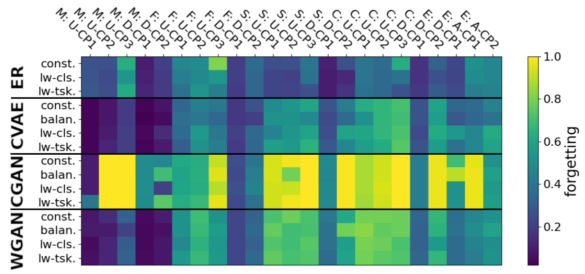
[50] Chenshen Wu, Luis Herranz, Xialei Liu, Yaxing Wang, Joost van de Weijer, and Bogdan Raducanu. Memory replay gans: learning to generate images from new categories without forgetting, 2019. 3

[51] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 4

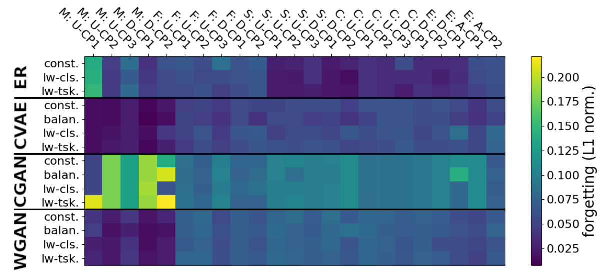
An analysis of best-practice strategies for replay and rehearsal in continual learning

Supplementary Material

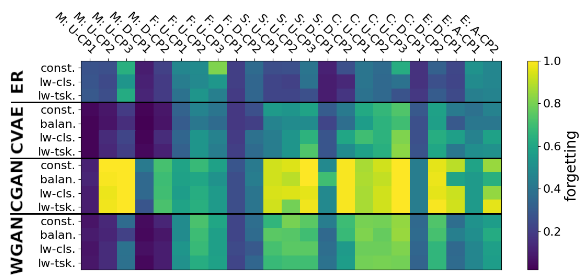
6. Rationale



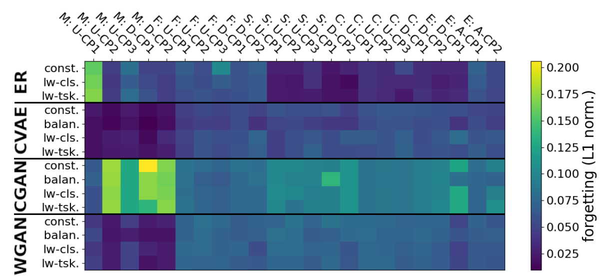
(a) Absolute forgetting (warm-start).



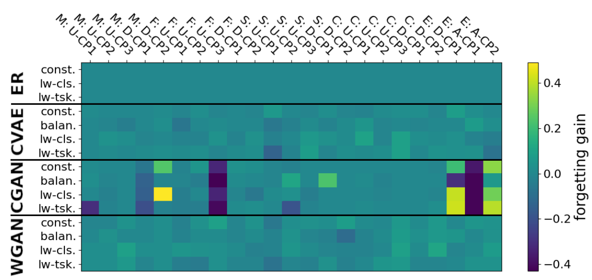
(b) Forgetting normalized over each column vector (warm-start).



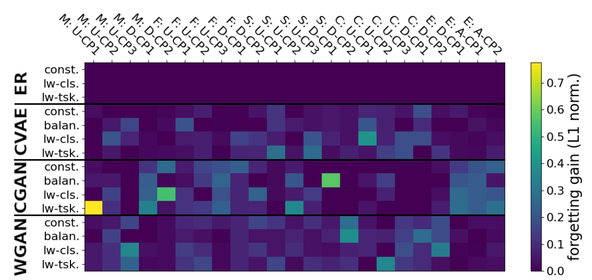
(c) Absolute forgetting (generator reset).



(d) Forgetting normalized over each column vector (generator reset).



(e) Difference in absolute forgetting between (c.) and (a.).



(f) Forgetting in normalized accuracy between (c.) and (a.).

Figure 4. This illustration showcases the averaged forgetting $F_{T_{\text{end}}}$ for all investigated datasets/splits. Each column represents a distinct task split on each dataset, whereas the first letter ("M", "F", "S", "C" stands for MNIST, Fashion-MNIST, SVHN and CIFAR-10), followed by an task split descriptor from Tab. 1. The deployed CL methods (4 groups * N rows) were modified as explained in Sec. 2.3: *const.* = constant, *balan.* = *balanced* (DGR exclusive), *lw-cls.* = loss-weighting by class count, *lw-tsk.* = loss-weighting by task count. The results for ER (top-most three rows) are re-used and serve as a baseline for the results in (c.-f.).