Adiabatic replay for continual learning

1st Anonymous Author
Anonymous Department
Anonymous Affiliation
Nowhere
anonymous email

2nd Anonymous Author
Anonymous Department
Anonymous Affiliation
Nowhere
anonymous email

***Abstract*—To avoid catastrophic forgetting, many replay-based approaches to continual learning (CL) require, for each learning phase with new data, the replay of samples representing *all* of the previously learned knowledge. Since this knowledge grows over time, such approaches invest linearly growing computational resources just for re-learning what is already known. In this proof-of-concept study, we propose a generative replay-based CL strategy that we term adiabatic replay (AR), which achieves CL in constant time and memory complexity by making use of the (very common) situation where each new learning phase is *adiabatic*, i.e., represents only a small addition to existing knowledge. The employed Gaussian Mixture Models (GMMs) are capable of *selective updating* only those parts of their internal representation affected by the new task. The information that would otherwise be overwritten by such updates is protected by *selective replay* of samples that are similar to newly arriving ones. Thus, the amount of to-be-replayed samples depends not at all on accumulated, but only on added knowledge, which is small by construction. Based on the challenging CIFAR and SVHN datasets in combination with pre-trained feature extractors, we confirm AR's superior scaling behavior while showing better accuracy than common baselines in the field.**

***Index Terms*—catastrophic forgetting, continual learning, class-incremental learning, generative replay, selective replay, selective updating, adiabatic replay**

I. Introduction

This contribution is in the context of continual learning (CL), a recent flavor of machine learning that investigates learning from data with non-stationary distributions. A common effect in this context is catastrophic forgetting (CF), an effect where previously acquired knowledge is abruptly lost after a change in data distributions. In class-incremental CL (see, e.g., [1], [2]), a number of assumptions are made: distribution changes are assumed to be abrupt, partitioning the data stream into stationary *tasks*. Then, task onsets are supposed to be known, instead of inferring them from data. Lastly, tasks are assumed to be disjoint. Together with this goes the constraint that no, or only a few, samples may be stored. A very promising approach to mitigate catastrophic forgetting (CF) in this scenario are replay strategies [3]. *Replay* aims at preventing CF by using samples from previous tasks to augment the current one. On the one hand, there are "true" replay methods which use a small number of stored samples for augmentation. On the other hand, there are *pseudo-replay* methods, where the samples to augment the current task are generated in potentially unlimited quantity, which removes the need to store samples. A schematics of the training process in generative replay is given in Fig. 1. Replay, in its original formulation, proposes a principled approach to CL, but it nevertheless presents several challenges: First of all, if DNNs are employed as solvers and generators, then all classes must be represented in equal proportion at every task in order to have any kind of performance guarantees. Thus, for example, in the simple case of a single new class (D samples) per task, the generator must produce $(s-1)D$ samples at task T_s in order to always train with D samples per class. This unbounded linear growth of to-be-replayed samples, and therefore of training time, as a function of the number of previous tasks s, poses enormous problems for long-term CL. For instance, even very small additions to a large body of existing knowledge (a common use-case) require a large amount of samples to be replayed, see, e.g., [4]. As replay is a lossy process, there are strict limits to GR-based CL performance.

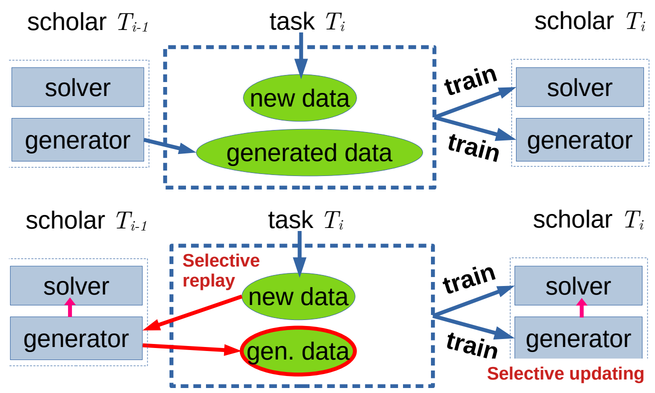

Fig. 1: Top: schematics of generative replay. A *scholar* composed of generator and solver is trained at every task. The solver performs the task, e.g., classification, whereas the generator serves as a memory for samples from previous tasks $T_{i'}$, $i' < i$. Please note that the amount of generated data usually far exceeds the amount of new data. Bottom: schematics of adiabatic replay, red indicates differences to generative replay. At every task, new data is used to query the generator, therefore generated data are produced in constant proportion. Furthermore, the generator additionally serves a feature generator for the solver, thus saving computational resources.

A. Approach: AR

Adiabatic replay (AR) prevents an ever-growing number of replayed samples by applying two main strategies: selective replay and selective updating, see Fig. 1. While the former means that new data are used to query the generator for similar (potentially conflicting) samples, the latter allows to only adapt specific regions of the semi-localized knowledge representation. For achieving selective replay, we rely on Gaussian Mixture Models (GMMs) trained by SGD as introduced in [5]. GMMs may have limited modeling capacity, but are sufficient when working with pre-trained feature extractors.

AR is partly inspired by maximally interfered retrieval (MIR), proposed in [6] where a fixed replay budget (either for experience replay or generative replay) is composed of the most conflicted samples, those that would be unlearned most rapidly when training on a new task. In a similar vein, [7] hypothesize that just replaying samples that are similar to new ones could be sufficient to avoid forgetting. Another inspiration comes from [8], where it is shown that replaying the right data at the right moment is preferable to replaying everything.

Adiabatic replay is most efficient when each task t, with data $\vec{x} \sim p^{(t)}$, adds only a small amount of new knowledge. AR models the joint distribution of past tasks as a mixture model with K components, $p^{(1\dots t-1)}(\vec{x}) = \sum_k \pi_k \mathcal{N}(\vec{x}; \vec{\mu}_k, \mathbf{\Sigma}_k)$, thus we can formalize this assumption as a requirement that only a few components in this mixture model are activated by new data: $|\{\arg\max_k \pi_k \mathcal{N}(\vec{x}_i; \vec{\mu}_k, \mathbf{\Sigma}_k), \vec{x}_i \sim p^{(t)}\}| \ll K$. A violation of this assumption would not necessarily break AR, but more components will need to be updated, requiring more samples. It was demonstrated in [5] that GMMs have an intrinsic capacity for *selective updating* when re-trained with new data. Concretely, only the components that are similar to, and thus potentially in conflict with, incoming data are adapted. In contrast, dissimilar components are not adapted, and are thus protected against CF.

B. Contributions

Selective replay: Previous knowledge is not replayed indiscriminately, but only where significant overlap with new data exists.

Selective updating: Previous knowledge is only modified by new data where an overlap exists.

Near-Constant time complexity: Assuming that each task adds only a small fraction to accumulated knowledge (adiabatic assumption), the number of generated/replayed samples can be small as well, and in particular does not grow with the number of tasks.

Integration of pre-trained feature extractors: To process visual problems of higher complexity (SVHN, CIFAR), we incorporate recent advances in latent replay into AR, where we do not replay raw samples but higher-level representations generated by a frozen feature extractor network.

C. Related Work

In recent years, many methods from a broad spectrum for mitigating CF and enabling CL have been presented, please refer to [9]–[12] for an overview. In this paper, however, we will focus on rehearsal-type CL.

Rehearsal/Replay Such solutions rely on the storage of previously encountered data instances, and may follow quite simplistic but still very effective techniques to avoid CF, i.e. by mixing real data with the content of some saved buffer, as shown in [13]–[16]. This has drawbacks in practice, since it breaks the constraints for task-incremental learning [17], has privacy concerns, and requires significant memory. *Partial replay*, e.g. [18], and constraint-based optimization [19]–[22], aims to select a subset for replay, but it appears that appropriate sample selection is still challenging [23]. Comprehensive overviews about current advances in replay can be found in [24], [25].

Deep Generative Replay Here, (deep) generative models are used for memory consolidation by replaying samples from previous tasks, see Fig. 1 and [26]. The recent growing interest in GR brought up a variety of architectures, building up on either VAEs [27]–[32] or GANs [33]–[36]. Notably, the MerGAN model [37] uses an LwF-type knowledge distillation technique to prevent forgetting in generators, which is more efficient than pure replay. Furthermore, PASS [38] uses self-supervised learning by sample augmentation in conjunction with slim class-based prototype storage for improving the performance replay-based CL. An increasingly employed technique in this respect is *latent replay* which operates on and replays latent features generated by a frozen encoder network, see, e.g., [3], [39], [40]. Built on this idea are models like REMIND [41], which extends latent replay by the aspect of compression, or SIESTA [42] which improves computational efficiency by alternating wake and sleep phases in which different parts of the architecture are adapted.

MIR Conceptually, this is similar to the concept of selective replay, although a key difference is that AR's GMM-generator and solver are capable of selective updating as well. We will use MIR as one of the baselines for latter experiments.

II. Methods

In the experimental part of this study we investigate adiabatic replay (AR), experience replay (ER), deep generative replay (DGR), brain-inspired replay (BI-R), maximally interfered retrieval (MIR) as well as pre-trained feature extractors.

A. Adiabatic replay (AR)

In contrast to original replay, where a scholar instance is composed of two complementary networks, namely a generator and solver, see Fig. 1, AR proposes a single network acting as a conventional generator, as well as a feature generator for the solver. Assuming a suitable scholar (see below), the high-level logic of AR is showcased in Fig. 2: Each sample from a new task is used to *query* the scholar, which generates a similar, known sample. Mixing new and generated samples in a predefined constant proportion, creates the training data for the

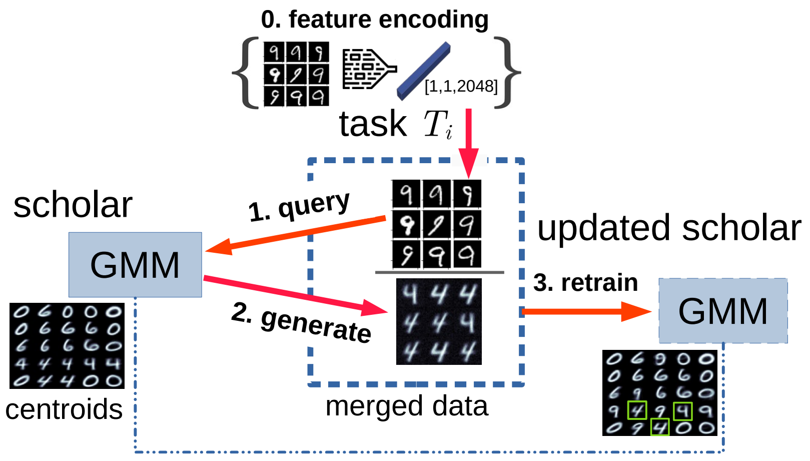

Fig. 2: The proposed AR approach, illustrated in an exemplary MNIST setting. The scholar (GMM) has been trained on MNIST classes 0, 4 and 6 in a previous task T_1. At task T_2, new data (class 9) is used to *query* the scholar for similar samples, resulting in the selective replay of mostly 4's but no 0's nor 6's. The scholar is re-trained *from its current state*, so no data concerning class 0 and 6 are required. Re-training results in the insertion of 9's into the existing components while still protecting some of the components representing 4's (green boxes). This mechanism works identically for higher-level features produced by a pre-trained feature extractor, see the optional step 0.

current task (see Algorithm 1 for pseudocode). A new sample will cause adaptation of the scholar in a localized region of data space. Variants generated by that sample will, due to similarity, cause adaptation in the same region. Knowledge in the overlap region will therefore be adapted to represent both, while dissimilar regions stay unaffected (see Fig. 2 for a visual impression). None of these requirements are fulfilled by DNNs, which is why we implement the scholar by a GMM layer (generator/feature encoder) followed by a linear classifier (solver). Both are independently trained via SGD according to [5].

```
Data: AR scholar/gen. $\Phi$, AR solver $\Theta$, real data $\mathcal{X}^t$, $Y^t$
for $t \in 2...T$ do
    for $\mathcal{B}_N \sim \mathcal{X}^t$ do
        // Propagate batch $\mathcal{B}_N$ though $\Phi$.
        $\sigma_{\mathcal{B}_N} \leftarrow \Phi(\mathcal{B}_N)$;
        // Query batch of variants from $\Phi$.
        $\mathcal{B}_G \leftarrow Vargen(\Phi, \sigma_{\mathcal{B}_N})$ ;
        // Add gen. samples to $\mathcal{X}^t_G$.
        $\mathcal{X}^t_G \leftarrow UpdateData(\mathcal{B}_G)$
    end
    for $\mathcal{B}_M \sim (\mathcal{X}^t \cup \mathcal{X}^t_G)$ do
        // Update $\Phi$ and $\Theta$
        $\Phi \leftarrow SGD(\mathcal{B}_M)$;
        $\Theta \leftarrow SGD(\Phi(\mathcal{B}_M), Y_t)$;
    end
end
```

Algorithm 1: Adiabatic Replay

Selective updating is an intrinsic property of GMMs. They describe data distributions by a set of K *components*, consisting of component weights π_k, centroids $\boldsymbol{\mu}_k$ and covariance matrices $\boldsymbol{\Sigma}_k$. A data sample $\boldsymbol{x}$ is assigned a probability $p(\boldsymbol{x}) = \sum_k \pi_k \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ as a weighted sum of normal distributions $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. Training of GMMs is performed as detailed in [5] by adapting centroids, covariance matrices and component weights through the SGD-based minimization of the negative log-likelihood $\mathcal{L} = \sum_n \log \sum_k \pi_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. As shown in [5], this expression is strongly dominated by a single GMM component k^*, and can be approximated as $-\log(\pi_{k^*} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{k^*}, \boldsymbol{\Sigma}_{k^*}))$. This implies that the best-matching GMM component k^* is the only component that selectively adapted.

Selective replay is a form of sampling from the probability density represented by a trained GMM, see [43]. It is triggered by a query in the form of a data sample $\boldsymbol{x}_n$, which is converted into a control signal $\mathcal{T}$ defined by the posterior probabilities (or *responsibilities*):

$$\gamma_k(\boldsymbol{x}_n) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{1}$$

For selective replay, these responsibilities parameterize a multinomial distribution for drawing a GMM component k^* to sample from, instead of the component weights π_K as usually done in GMM sampling. To reduce noise, top-S sampling is introduced, where the three highest component responsibilities determine the selection.

Solver functions are performed by feeding GMM responsibilities into a linear regression layer as $\boldsymbol{o}(\boldsymbol{x}_n) = \boldsymbol{W}\boldsymbol{\gamma}(\boldsymbol{x}_n)$. We use a MSE loss and drop the bias term to reduce the sensitivity to unbalanced classes.

III. Experimental setup

A. Data

MNIST [44] consists of 60.000 28×28 grayscale images of handwritten digits (0-9).

Fashion-MNIST [45] consists of 60.000 images of clothes in 10 categories and is structured like MNIST.
E-MNIST [46] is structured like MNIST and extends it by letters. We use the balanced split which contains 131.000 samples in 47 classes.
SVHN [47] contains 60.000 RGB images of house numbers (0-9, resolution 32×32).
CIFAR-10 [48] contains 60.000 RGB images of natural objects, resolution 32x32, in 10 balanced classes.
Feature encoding of SVHN and CIFAR is described in Sec. IV-F. However, for MNIST, Fashion-MNIST and E-MNIST it is not performed due to their inherent simplicity.
Class incremental learning (CIL) problems are constructed by splitting the datasets as follows: D7-1^3A (4 tasks, 0-6,7,8,9), D7-1^3B (4 tasks, 3-9,0,1,2), D5-1^5A (6 tasks, 0-4,5,6,7,8,9), D5-1^5B (6 tasks, 5-9,0,1,2,3,4), D2-2^5A (5 tasks, 0-1,2-3,4-5,6-7,8-9), D2-2^5B (5 tasks, 8-9,6-7,4-5,2-3,0-1), D20-1^5A (6 tasks, 0-19,20,21,22,23,24, EMNIST only) and D20-1^5B (6 tasks, 5-24,0,1,2,3,4, EMNIST only). All data are normalized to a range of $[0, 1]$ before being processed.

B. Evaluation metrics

When training a solver $\mathcal{S}$ on a class-incremental learning problem with T tasks, let the accuracy metric α_{ij} denote the accuracy of the solver $\mathcal{S}_i$ after being trained on tasks $1, \ldots, i$ and evaluated on the test set of task j with $i, j < T$.

The final accuracy α_T is computed by evaluating $\mathcal{S}_T$ on a joint test set T_{ALL}. For reference, α^{base} measures the joint-training (JT) performance, i.e., the accuracy achieved on T_{ALL} by a scholar after training on the union of all tasks. Similar to [49], [50], we define an average forgetting measure F_T which reflects the loss of knowledge about previous tasks, and which is defined as follows:

$$F_{ij} = \max_{i \in \{1,..,T-1\}} \alpha_{ij} - \alpha_{Tj} \qquad \forall j < T$$
$$F_T = \frac{1}{T-1} \sum_{j=1}^{T-1} F_{Tj} \qquad F_T \in [0, 1]. \tag{2}$$

C. Scenarios for replay experiments

For all replay experiments (AR, ER and DGR), we distinguish three different experimental scenarios: constant, balanced and weighted.

In the *constant* scenario, replay of D_i samples is performed before training on task $T_i, i > 1$ using the current scholar S_{i-1}, where D_i represents the amount of training samples contained in T_i. This strategy keeps the amount of generated samples constant w.r.t the number of tasks, and thus comes with modest temporary storage requirements. Mini-batches β consist to an equal proportion of samples from T_i, as well as generated samples. It is worth noting that, in this scenario, classes will in general *not* be balanced, and that it is not required to store the statistics of previously encountered class instances/labels.

In the *balanced* scenario, the amount of replayed samples D_i scales linearly w.r.t the number of tasks, and the proportions of classes in a mini-batch are approximately balanced.

The *weighted* scenario is similar in spirit to the constant scenario, with an enhancement inspired by [3]. Here, the losses of the generator and solver networks consist of two terms: L_{current} (data from the current task T_i) and L_{replay} (generated/buffered data). Both loss terms are weighted to offset the fact that classes are unbalanced in generated and current data. Assuming that the amount of samples per class is roughly similar, we compute these weights according to the total number of classes N encountered in previous and current tasks $T_1 ... T_T$:

$$\mathcal{L}_{total} = \frac{1}{N_{T_1 \ldots T_{T-1}}} \mathcal{L}_{current} + \frac{N_T}{N_{T_1 \ldots T_{T-1}}} \mathcal{L}_{replay} \tag{3}$$

IV. Implementation details

A. AR

AR employs a GMM scholar $L_{(G)}$ with $K = 225$ components and diagonal covariance matrices. The choice of K is subject to a "the more the better" principle, and is limited only by available GPU memory. AR training consists of an (initial) run on T_1, followed by a sequence of independent (replay) runs on $T_i, i > 1$. Mini-batch size β is set to 100 for all experiments.

GMM generator training follows the procedures and best-practice settings presented and justified in [5], please refer to this work for recommended learning and regularization hyperparameters. However, it should be noted that their choice is less sensitive to any particular CL problem at hand.

Training is terminated via early stopping when $L_{(G)}$ reaches a plateau of stationary loss for the current task T_i. We set the training epochs to 512 as an upper bound. Both, $L_{(G)}$ and the classification head are independently optimized via vanilla SGD using a fixed learning rate of $\epsilon = 0.05$. The relative strengths of component weight and covariance matrix adaptation are set to 0.1. Annealing control regulates the component adaptation radius for $L_{(G)}$, and sets $\sigma \leftarrow 0.96\sigma$ whenever the loss is considered sufficiently stationary. The initial value is set to $\sigma_0^{init} = \sqrt{0.125K}$ for the first task T_1, and $\sigma_0^{replay} = 0.1$ for subsequent (replay) tasks $T_i, i > 1$. GMM sampling parameters $S = 3$ (top-S) and $\rho = 1.0$ (normalization) are kept fixed throughout all experiments.

B. DGR

Deep generative replay (DGR) is implemented using VAEs as generators. We choose VAEs over GANs or WGANs due to the experiments conducted in [4], which suggest that GANs require extensive structural tuning, which is by definition excluded in a CL scenario for all tasks but the first. Similarly, GANs and VAEs were both used in CL research, e.g., in [51] with comparable performance.

The network structure of the generator and solver is given in Tab. I. Please note that encoders/decoders are

Encoder (DGR)	C2D(32,3,2)-ReLU-C2D(64,3,2)-ReLU-Dense(512)-ReLU-Dense(256)-ReLU-Dense(z*2)
Decoder (DGR)	Dense(256)-ReLU-Dense(512)-ReLU-Dense((H/4)*(W/4)*64)-ReLU-Reshape((H/4),(W/4),64)-ReLU-C2DTr.(32,3,2)-ReLU-C2DTr.(C,5,2)-Sig. C2DTr.(64,3,2)-ReLU-
Encoder (LR)	Dense(2000)-ReLU-Dense(2000)-ReLU-Dense(z*2)
Decoder (LR)	Dense(128)-ReLU-Dense(512)-ReLU-Dense(1024)-ReLU-Dense(H*W*C)
Solver	Dense(400)-ReLU-Dense(400)-ReLU-Dense(400)-ReLU-Dense(N)-SoftMax
Solver (ER)	C2D(32,3,2)-ReLU-MaxPool2D(2,2)-C2D(64,3,2)-ReLU-MaxPool2D(2,2)-Dense(512)-ReLU-Dense(128)-ReLU-Dense(N)-SoftMax

TABLE I: ANN structures for DGR-VAE/ER. Encoder/decoder is exclusive for DGR and represents the employed VAE generator. For latent replay, the intermediate layers of the generator (DGR) and solver (DGR/ER) up to output layers (Gaussians/Softmax) were replaced with fully-connected layers and ReLU activation.

fully-connected DNNs, when operating on latent features (SVHN, CIFAR). The latent dimension z is set to 100 and the ELBO loss uses a disentangling factor of $\beta = 1.0$. Label information is incorporated to condition the latent space to preserve the generator's ability to generate equal proportions for previously seen classes. The learning rate for VAE generators and solvers are set to $\epsilon_G = 10^{-4}$, $\epsilon_S = 10^{-3}$ using the ADAM optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$. Generators and solvers are trained for 200 and 50 epochs, respectively.

C. ER

ER uses a solver with the layer structure shown in Tab. I, see Solver (ER). The ADAM optimizer is used with a learning rate of 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and the scholar is trained for 100 epochs on each task. Analogous to the procedure for DGR, we use replay on latent feature representations, see e.g., [39] as described in Sec. IV-F for SVHN and CIFAR. In this case, the network is taken to be the same as the solver for DGR. Reservoir sampling [52] is used to select 50 samples of each encountered class to be stored per task. For replay, oversampling of the buffer is performed to obtain a number of samples, equal to the amount of data instances D_i present in the current task T_i.

D. Gen-MIR

We perform Gen-MIR experiments as described in [6] using the author's software, adapted to work with different datasets and dataset splits. We employ the optimal parameter settings for the Split-MNIST problem as described in [6], using 5500 training samples per task. On top of that, we perform a grid search in parameter space for the parameters n_mem $\in \{1,3\}$, gen_iters $\in \{5,15\}$, cls_iters $\in \{5,15\}$, mir_iters $\in \{2,5,10\}$, gen_kl_coeff $\in \{0.1,0.5,1.0\}$, mem_coeff $\in \{1,3\}$. Since the first task for the EMNIST dataset contains a large number of classes, we use 10.000 samples in this case.

E. BI-R

Brain-inspired replay experiments are conducted as described in [3], using the author's code adapted to different datasets and dataset splits. Best-practice parameters as described by the authors are used, except that we modify the weighting of present and past tasks based on the number of classes in past and current tasks, see Sec. III-C.

F. Use of pre-trained feature extractors

It has been shown that encoding features using pre-trained networks, which convert raw data into a higher quality and more invariant representation, is beneficial for CL [3], [39], [41]. A current promising direction of pre-training such models is *contrastive learning*, which is performed in a supervised [53] (SupCon) or self-supervised fashion [54]–[56] (SSCL). In this study, we rely on SupCon to build a robust feature extractor for more complex datasets (SVHN, CIFAR).

We take a portion of the data from the target domain for pre-training, but exclude these instances from further usage in downstream CL tasks. For SVHN, we pull an amount equal to 0.5 of the total training samples from the ”extra” split. For CIFAR10 we split the training set in half and use one for pre-training and the other for encoding and later usage in downstream CL. The data used to pre-train the feature extractor are thus similar but not identical to subsequent training data, following the approach of [3]. An additional data augmentation module normalizes the input, performs random horizontal flipping and rotation in the range of $-2\% * 2\pi - +2\% * 2\pi$ for each input image. The encoder backbone is a ResNet-50 with randomly initialized weights and is trained for 256 epochs using a batch size of $\beta = 256$. No further fine-tuning is performed after pre-training. We use the normalized activations of the final pooling layer ($D = 2048$) as the representation vector. For supervised training, a projection head is attached, consisting of two hidden layers, having a total of 2048 and 128 projection units, followed by ReLU activation. The multi-class npairs loss [57] uses a temperature of 0.05 and is optimized via ADAM with a learning rate of $\epsilon = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. After pre-training we push the complete training data through the encoder network and save the output to disk for later usage. However, it would be perfectly legitimate to use the model on-the-fly to encode the data mini-batch wise, though this comes at the cost of a worse runtime efficiency.

G. Sequential Fine-Tuning (SFT)

For SFT, we use the baseline DNN which is identical to the ER solver. This DNN is trained with a normal learning rate of 0.001 on T_1, and with a reduced learning rate of 0.0001 for all subsequent tasks.

V. Experiments and Results

All experiments are run on a cluster of 30 machines equipped with single RTX3070Ti GPUs. We perform ten randomly initialized runs for the supervised CIL problems described in Sec. III-A, as well as baseline experiments to measure offline JT performance.

A. Selective replay functionality

First, we demonstrate the ability of a trained GMM to query its internal representation through data samples and selectively generate artificial data that ”best match” those defining the query. To illustrate this, we train a GMM layer of $K = 25$ components on MNIST classes 0, 4 and 6 for 50 epochs using the best-practice rules described in Sec. IV-A. Then, we query the trained GMM with samples from class 9 uniquely, as described in Sec. II. The resulting samples are all from class 4, since it is the class that is ”most similar” to the query class. These results are visualized in Fig. 2. Variant generation results for deep convolutional extensions of GMMs can be found in [58], emphasizing that the AR approach can be scaled to more complex problems. This approach works identically for latent replay, as it only requires prior encoding of the raw data, as shown in the figure.

B. CIL comparison

This experiment evaluates the CL performance of AR w.r.t. the measures given in Sec. III-B, and compares its performance to MIR, BI-R, DGR-VAE, and ER (see Sec. IV), since these represent principled approaches to replay-based CL. Further, we show results for sequential fine-tuning (SFT), as well as baseline joint training performance on all datasets for a DNN (identical to the ER solver) and AR. The results are given in Tab. II.

Baseline and initial task performance We observe superior JT (i.e., non-CL) performance for the DNN on all datasets, except encoded CIFAR as shown in Tab. II (bottom part). This indicates that the DNN scholar is better suited for the discrimination of all investigated ”raw” datasets. This advantage diminishes for latent replay, however. On the other hand, AR relies on a considerably less complex structure in its current state. For instance, DGR uses significantly more trainable parameters, especially when operating on latent features, or generally speaking, data of high dimensionality.

Th factor is 8.7 when applying DGR to RGB data and 16 when applying it to latent features. The ability to perform well on the JT may also directly translate to a better starting point for CIL with DGR and ER due to the higher performance reached on the initial task T_1.

Constant-time replay is problematic for DGR We observe that DGR performs worse in the *constant* setting, at least for data with higher complexity than MNIST, or when the number of tasks increases. This observation can be confirmed by experiments with different numbers of tasks. To a slightly lesser extent, this is also observed for ER on, e.g., EMNIST. In contrast, AR is specifically designed to work well when the amount of generated samples is kept constant for an increasing number of tasks. Figure 3 shows the count of generated samples over time for AR and DGR in a balanced scenario for MNIST D5-1^5A, respectively. This also inevitably results in longer training periods for the VAE generator.

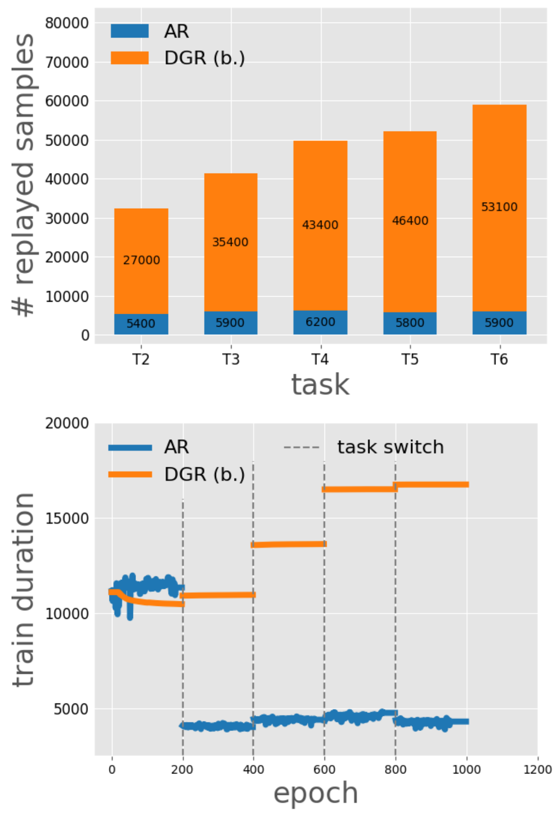

Fig. 3: Top: The amount of samples generated per task for the MNIST D5-1^5 problem by DGR if class balancing is to be achieved, and a comparison to samples generated by AR. Bottom: The averaged per epoch training duration of DGR (b.) and AR. Dashed lines indicate a task switch.

AR –vs– ER Generally, ER shows good results, stable forgetting rates and often outperforms AR when operating on raw inputs (MNIST, FMNIST and E-MNIST), although the differences are not striking. However, a comparison between ER and AR is biased in favor of ER since AR does not get to see any real samples from past tasks. Rather, ER serves as a baseline of what can be reasonably expected from AR, and we observe that this baseline is generally quite well egalized. It is surprising to see that latent AR is able to achieve generally better results than latent ER. It could be argued that the budget per class for a more complex dataset like SVHN and CIFAR-10 is rather small, and it can be assumed that increasing the budget would increase CL performance. However, we reiterate that this is not trivially applicable in scenarios with a constrained memory budget.
Furthermore, ER has the disadvantage that training time and memory usage grow slowly but linearly with each added task, which is a unrealistic premise in practice. A fixed memory budget mitigates this problem, but has the negative effect that samples from long-ago tasks will be lost over time, which will render ER ineffective if the number of tasks is large.

	CIL-P	D7-1^3A				D7-1^3B				D20-1^5A	D20-1^5B
	dataset	MNIST	F-MNIST	SVHN	CIFAR10	MNIST	F-MNIST	SVHN	CIFAR10	E-MNIST	
method	AR	.75 / .10	.70 / .10	**.89** / **.03**	**.70** / **.05**	.83 / .07	.68 / **.10**	**.91** / **.02**	**.64** / **.17**	.53 / **.10**	.53 / **.09**
	BI-R (w.)	.75 / .11	.71 / .12	.58 / .18	.52 / .19	.77 / .09	**.70** / .12	.55 / .19	.49 / .22	.0.25 / .42	.0.27 / .39
	DGR (b.)	**.93** / **.03**	**.80** / **.08**	.63 / .18	.51 / .18	.93 / **.03**	**.70** / .16	.73 / .14	.45 / .25	.55 / .27	.62 / .26
	DGR (c.)	**.93** / **.04**	.78 / .10	.52 / .25	.44 / .33	**.94** / **.03**	**.70** / .17	.59 / .20	.39 / .33	.61 / .28	.57 / .22
	DGR (w.)	.83 / .12	.75 / .18	.57 / .19	.42 / .22	.91 / .04	.66 / .28	.59 / .20	.37 / .35	.43 / .79	.45 / .61
	ER (c.)	.90 / .08	.77 / .13	.76 / .10	.59 / .13	.92 / **.03**	.68 / .18	.78 / .10	.54 / .22	**.78** / .22	**.78** / .20
	ER (w.)	.90 / .07	.78 / .13	.80 / .09	.61 / .12	.92 / **.03**	.68 / .18	.79 / .10	.57 / .20	.75 / .21	.77 / .19
	MIR (c.)	.79 / .13	.72 / .15	.58 / .19	.53 / .23	.78 / .10	**.70** / .11	.53 / .19	.45 / .24	.44 / .26	.46 / .28
	MIR (w.)	.78 / .11	.73 / .12	.58 / .18	.55 / .21	.71 / .13	.65 / .16	.60 / .18	.44 / .23	.49 / .25	.51 / .25
	SFT	.18 / .55	.13 / .51	.14 / .55	.15 / .54	.14 / .56	.11 / .53	.13 / .53	.11 / .57	.05 / .56	.04 / .57

CIL-P	D2-2^4A				D2-2^4B				D5-1^5A				D5-1^5B			
dataset	MNIST	F-MNIST	SVHN	CIFAR10	MNIST	F-MNIST	SVHN	CIFAR10	MNIST	F-MNIST	SVHN	CIFAR10	MNIST	F-MNIST	SVHN	CIFAR10
AR	.69 / .21	.61 / .36	**.77** / .22	.61 / .35	.74 / .14	.63 / .35	**.93** / **.07**	**.68** / .35	.71 / .17	.64 / .27	**.87** / **.09**	**.64** / **.17**	.73 / .10	.67 / .33	**.86** / **.08**	.60 / **.19**
BI-R (w.)	.84 / .08	**.78** / **.11**	.56 / **.16**	.52 / **.19**	.83 / .08	.75 / **.13**	.58 / .18	.49 / **.22**	.79 / .13	.75 / **.14**	.65 / .17	.61 / .19	.80 / .09	.72 / **.13**	.64 / .16	.59 / .20
DGR (b.)	**.94** / **.07**	.73 / .39	.58 / .42	.45 / .61	.92 / .09	.73 / .32	.75 / .35	.42 / .68	.88 / .09	.72 / .27	.55 / .39	.42 / .48	**.89** / **.06**	.71 / .34	.66 / .38	.39 / .58
DGR (c.)	.93 / .08	.71 / .44	.48 / .52	.41 / .63	**.94** / **.07**	.72 / .35	.63 / .51	.40 / .69	**.90** / **.08**	.74 / .27	.46 / .43	.41 / .52	**.89** / .07	.68 / .37	.60 / .45	.37 / .65
DGR (w.)	.86 / .17	.70 / .49	.52 / .54	.39 / .65	.91 / .10	.71 / .38	.62 / .54	.36 / .70	.75 / .30	.68 / .46	.47 / .55	.35 / .62	.84 / .14	.63 / .53	.49 / .58	.34 / .74
ER (c.)	.86 / .17	.66 / .58	.74 / .27	.60 / .46	.88 / .17	.67 / .48	.80 / .27	.58 / .49	.87 / .16	.73 / .39	.76 / .25	.59 / .33	**.89** / .09	.68 / .42	.77 / .26	.56 / .48
ER (w.)	.87 / .17	.65 / .57	**.77** / .24	.59 / .44	**.89** / .17	.67 / .47	.83 / .24	.61 / .45	.86 / .16	.72 / .37	.78 / .22	.59 / .34	**.89** / .10	.69 / .41	.79 / .23	.59 / .41
MIR (c.)	.81 / .10	.76 / .12	.71 / .23	**.65** / .35	.80 / .09	.71 / .24	.70 / .25	.63 / .31	.79 / **.08**	.76 / **.14**	.68 / .20	.59 / .33	.78 / .14	.74 / .17	.66 / .24	.57 / .29
MIR (w.)	.83 / .08	.75 / .12	.74 / **.16**	.63 / .25	.82 / .09	**.77** / .14	.77 / .14	.65 / .23	.80 / .13	**.78** / .16	.73 / .26	.58 / .41	.81 / .13	**.75** / .16	.72 / .18	**.61** / .27
SFT	.18 / .55	.12 / .61	.25 / .58	.11 / .61	.20 / .51	.15 / .56	.13 / .61	.11 / .60	.16 / .62	.15 / .64	.11 / .65	.11 / .61	.17 / .54	.13 / .54	.12 / .63	.12 / .60

	DS	MNIST	F-MNIST	SVHN	CIFAR-10	E-MNIST
method	AR	.92	.78	**.92**	**.75**	.67
	ANN	**.99**	**.89**	**.92**	.71	**.88**

TABLE II: Experimental results. **Upper two tables** Results of all investigated methods for each CIL-P. Certain methods are deployed under specific replay scenarios (b. = balanced, c. = constant-time, w. = weighted sample loss). We present the final test-set accuracy α_T followed by the forgetting measure F_T. **Lower table** The joint-training baseline for all datasets. As a model, we use the ER solver from Tab. I which is trained for 100 epochs. AR uses the architecture and training scheme as described in Sec. IV-A. Results are always averaged across $N = 10$ runs. Detailed information about the evaluation process and experimental setup can be found in Sec. III-B.

AR –vs– Gen-MIR and BI-R MIR and AR share the concept of selective replay, and they both operate in a constant-time scenario although MIR can weight generated and new samples differently in the loss, and thus is conducted within the *weighted* scenario. MIR is highly sensitive to parameters like the weights of different terms in the loss, which must be set by cross-validation and are thus strictly speaking not compatible with CL. The results also show that MIR performs less well for the latent datasets CIFAR and SVHN. We speculate that this is due to the online setting which allows only a limited number of epochs to be processed per task, which could be insufficient for VAE training on complex data. However, we found that if we increased the number of epochs too much, forgetting of old tasks was strongly amplified. This could be due to the fact that Gen-MIR implements selective replay but cannot perform selective updates. Similar comments apply to BI-R, which shows strongly reduced performance on the latent replay tasks. As a side note, AR incorporates two key concepts from BI-R (conditional generation and latent replay), although it does not rely on VAEs for replay, which is presumably beneficial for performance.

AR: sequential fine tuning (SFT) It is not entirely surprising that SFT results are inferior throughout, since SFT implements no dedicated mechanisms to prevent CF. While SFT may be a viable method when the number of tasks is very small, it is clear from the presented results that this is no longer the case as the number of tasks increases.

AR: selective replay AR shows promising results in terms of knowledge retention and preventing CF for sequentially learned classes, as reflected by generally lower average forgetting scores. In virtually all of the experiments, we observe a very moderate loss of knowledge of the first task T_1 after full training, even if T_1 is large such as for the EMNIST experiments. This suggests that AR's ability to handle small incremental additions/updates to the internal knowledge base over a sequence of tasks is an intrinsic property, due to the selective replay mechanism. Moreover, AR demonstrates its intrinsic ability to limit unnecessary overwrites of past knowledge by performing efficient *selective updates*, instead of having to replay the entire accumulated knowledge each time a task is added.

AR: selective updates As performed by AR training, are mainly characterized by matching GMM components with arriving input. Therefore, performance on previous tasks generally decreases only slightly by the adaptation of selected/similar units, as shown by the low forgetting rates for almost all CIL-P studied in Tab. II. This implies that the GMM tends to converge towards a *trade-off* between past knowledge and new data. This effect is most notable when there is successive (replay-)training for two classes with high similarity in the input space, such as with, F-MNIST: D5-1^5A, where task T_2 (class: "sandals") and task T_4 (class: "sneakers") compete for internal capacity.

VI. Discussion

In summary, we can state that our AR approach is able to compete with SOTA replay-based architectures for CL problems derived from simple datasets. In contrast, it outperforms all other methods for more complex datasets, combined with latent replay. The performance difference is particularly evident when the investigated methods were constrained to a constant-time replay setting and was not significantly altered by weighted replay. This is remarkable because the AR scholar performs the tasks of both solver and generator, while at the same time having less parameters than, e.g., DGR-VAE. We may therefore conclude that AR offers a principled approach to truly long-term CL. In the following text, we will discuss salient points concerning our evaluation methodology and the conclusions we draw from the results:

Data Some datasets are not considered meaningful benchmarks in non-continual ML due to their simplicity. Still, many CL studies rely on these two datasets, which is why they are included for comparison purposes. SVHN and CIFAR-10 in particular are considered challenging for generative replay methods, see [6]. E-MNIST represents a simple classification benchmark that is quite hard for CL due to the large number of classes and is well-suited to the targeted AR scenario where each task adds only a small fraction of knowledge.

Unbalanced tasks Typical class-incremental CL problems in the literature consist of tasks containing the same number of classes. The most commonly used CL task is what we term $D2$-2^4, i.e., two classes to any of 5 tasks. The fact that all tasks have the same amount of samples and classes is by now a more or less implicit assumption for many CL algorithms. For example, BI-R sets the task weights according to the number of tasks, regardless of their size and class composition (we adapted this algorithm in that respect for our experiments). Similarly, the number of replayed samples in Gen-MIR is always taken to be identical, however it should be impacted by task sizes if one task is significantly larger than others (we adapted this as well for the EMNIST experiments). This study is, to our knowledge, the first to study CL for strongly unbalanced tasks, which may be a very common setting in real-world CL and should be properly investigated for newly proposed CL algorithms.

Pre-trained feature extractors The use of pre-trained models is appealing in CL, since a lot of complexity can be "outsourced" to these models. As shown in [59], the effectiveness of feature extraction from a frozen pre-trained model relies on the relation between downstream and upstream tasks. There seems to be excellent agreement between the often-used combination of CIFAR and ImageNet, but does not extend to, e.g., the SVHN dataset without fine-tuning. Thus, we chose separate pre-trained models for each dataset that were optimized in a supervised fashion (SupCon) on similar but not identical data, following [3]. In contrast, self-supervised contrastive learning alleviates the need of a large labeled pre-training dataset but relies on the usage of large mini-batch sizes, as well as complex data augmentation pipelines [54]–[56]. We decided against such methods as they only show competitive results when combined with supervised fine-tuning on labeled data [60], or significantly increasing the total amount of classes seen in pre-training [61].

Issues with constant-time replay Instead of achieving balance between new and recalled/generated samples by a linear increase of the latter, many recently proposed replay approaches use only a fixed number S of generated or recalled samples per task. Balance is realized by a higher weight of past samples in the loss [6]. There are several issues with this: First of all, for a large number of tasks, each task will be less and less represented in S samples, making eventual forgetting inevitable, while weights for past samples grow higher and higher. Then, giving past samples a higher weight effectively increases the learning rate for these samples, which can break SGD if the weights are too high. Alternatively, the weight for the current samples can be *reduced* from its baseline value in some works [3], ultimately leading to low learning rates and thus long training times. And lastly, the loss weights are generally set post-hoc via cross-validation [6], [37], which is inadmissible for CL because it amounts to knowing all tasks beforehand. AR can use constant-time replay without weighting past samples due to the selective updating and selective replay properties.

Violation of AR assumptions The assumption that new tasks only add a small contribution is not a hard requirement, just a prerequisite for sample efficiency. Based on the formalization presented in Sec. I, its validity is trivial to verify by examining component activations of the GMM generator when faced with new data. Although we do not implement such a control strategy here, AR would simply need to replay more samples if contributions of new data should be large. However, the chances of this happening in practice are low, if the body of existing knowledge is sufficiently large.

Initial annealing radius tuning AR contains a few technical details that require tuning, like the initial annealing radius parameter r_0 when re-training with new task data. We used a single value for all experiments, but performance is sensitive to this choice, since it represents a trade-off between new data acquisition and knowledge retention. Therefore, we intend to develop an automated control strategy for this parameter to facilitate experimentation.

VII. Conclusion

We firmly believe that continual learning (CL) holds the potential to spark a new machine learning revolution, since it allows, if it could be made to work in large-scale settings on real-world data, the training of models over very long times, and thus with enormous amounts of data. To achieve this important milestone, CL research must, to our mind, imperatively focus on aspects of long-term feasibility, such as also targeted in domains like life-long learning. A related aspect is energy efficiency: in order to be accepted and used in practice, training by CL must be comparable to joint training in terms of training time (and therefore energy consumption). Only in this case can the considerable advantages of CL be made to have beneficial effects in applications. In this study, we show a proof-of-concept for CL that makes a step in this direction. Namely, AR operates at a time complexity that is independent of the amount of previously acquired knowledge, which is something we also observe in humans. The overall time complexity is thus comparable to joint training. Further work will focus on the optimization of AR w.r.t. efficiency and ease of use, and the question of how to train the feature extractors in a continual fashion as well.

A. Reproducibility

We will provide a publicly available TensorFlow 2 implementation which will be made publicly available for the camera-ready version. This repository will contain step-by-step instructions to conduct the experiments described in this article. Additional details about experimental procedures and used parameter settings are given in the various sections of the appendix (after the references) which are referenced in the text.

References

[1] B. Bagus, A. Gepperth, and T. Lesort, “Beyond supervised continual learning: a review,” *arXiv preprint arXiv:2208.14307*, 2022.

[2] G. M. van de Ven, T. Tuytelaars, and A. S. Tolias, “Three types of incremental learning,” *Nature Machine Intelligence*, vol. 4, no. 12, pp. 1185–1197, 2022.

[3] G. M. van de Ven, H. T. Siegelmann, and A. S. Tolias, “Brain-inspired replay for continual learning with artificial neural networks,” *Nature communications*, vol. 11, no. 1, pp. 1–14, 2020.

[4] N. Dzemidovich and A. Gepperth, “An empirical comparison of generators in replay-based continual learning,” in *European Symposium on Artificial Neural Networks (ESANN)*, 2022.

[5] A. Gepperth and B. Pfülb, “Gradient-based training of gaussian mixture models for high-dimensional streaming data,” *Neural Processing Letters*, vol. 53, no. 6, pp. 4331–4348, 2021.

[6] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars, “Online continual learning with maximally interfered retrieval,” in *Neural Information Processing Systems*, 2019.

[7] J. L. McClelland, B. L. McNaughton, and A. K. Lampinen, “Integration of new information in memory: new insights from a complementary learning systems perspective,” *Philosophical Transactions of the Royal Society B*, vol. 375, no. 1799, p. 20190637, 2020.

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

[9] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, “A continual learning survey: Defying forgetting in classification tasks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

[10] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, “Class-incremental learning: survey and performance evaluation on image classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[11] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, “Embracing change: Continual learning in deep neural networks,” *Trends in cognitive sciences*, vol. 24, no. 12, pp. 1028–1040, 2020.

[12] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, “Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges,” *Information fusion*, vol. 58, pp. 52–68, 2020.

[13] A. Gepperth and C. Karaoguz, “A bio-inspired incremental learning architecture for applied perceptual problems,” *Cognitive Computation*, vol. 8, no. 5, pp. 924–934, 2016.

[14] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, “icarl: Incremental classifier and representation learning,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[15] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, “Experience replay for continual learning,” *Advances in Neural Information Processing Systems*, vol. 32, pp. 350–360, 2019.

[16] M. De Lange and T. Tuytelaars, “Continual prototype evolution: Learning online from non-stationary data streams,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8250–8259.

[17] G. M. Van de Ven and A. S. Tolias, “Three scenarios for continual learning,” *arXiv preprint arXiv:1904.07734*, 2019.

[18] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars, “Online continual learning with maximally interfered retrieval,” 2019. [Online]. Available: https://arxiv.org/abs/1908.04742

[19] D. Lopez-Paz and M. Ranzato, “Gradient episodic memory for continual learning,” *Advances in neural information processing systems*, vol. 30, pp. 6467–6476, 2017.

[20] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, “Efficient lifelong learning with a-gem,” *arXiv preprint arXiv:1812.00420*, 2018.

[21] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, “On tiny episodic memories in continual learning,” *arXiv preprint arXiv:1902.10486*, 2019.

[22] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, “Gradient based sample selection for online continual learning,” *arXiv preprint arXiv:1903.08671*, 2019.

[23] A. Prabhu, P. H. Torr, and P. K. Dokania, “Gdumb: A simple approach that questions our progress in continual learning,” in *European conference on computer vision*. Springer, 2020, pp. 524–540.

[24] T. L. Hayes, G. P. Krishnan, M. Bazhenov, H. T. Siegelmann, T. J. Sejnowski, and C. Kanan, “Replay in deep learning: Current approaches and missing biological elements,” *Neural Computation*, vol. 33, no. 11, pp. 2908–2950, 2021.

[25] B. Bagus and A. Gepperth, “An investigation of replay-based approaches for continual learning,” in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–9.

[26] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” *arXiv preprint arXiv:1705.08690*, 2017.

[27] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[28] N. Kamra, U. Gupta, and Y. Liu, “Deep generative dual memory network for continual learning,” *arXiv preprint arXiv:1710.10368*, 2017.

[29] F. Lavda, J. Ramapuram, M. Gregorova, and A. Kalousis, “Continual classification learning using generative models,” *arXiv preprint arXiv:1810.10612*, 2018.

[30] J. Ramapuram, M. Gregorova, and A. Kalousis, “Lifelong generative modeling,” *Neurocomputing*, vol. 404, pp. 381–400, 2020.

[31] F. Ye and A. G. Bors, “Learning latent representations across multiple data domains using lifelong vaegan,” in *European Conference on Computer Vision*. Springer, 2020, pp. 777–795.

[32] H. Caselles-Dupré, M. Garcia-Ortiz, and D. Filliat, “S-trigger: Continual state representation learning via self-triggered generative replay,” in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–7.

[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[34] O. Ostapenko, M. Puscas, T. Klein, P. Jahnichen, and M. Nabi, "Learning to remember: A synaptic plasticity driven framework for continual learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 321–11 329.

[35] L. Wang, K. Yang, C. Li, L. Hong, Z. Li, and J. Zhu, "Ordisco: Effective and efficient usage of incremental unlabeled data for semi-supervised continual learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5383–5392.

[36] C. Atkinson, B. McCane, L. Szymanski, and A. Robins, "Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting," *Neurocomputing*, vol. 428, pp. 291–307, 2021.

[37] C. Wu, L. Herranz, X. Liu, J. van de Weijer, B. Raducanu *et al.*, "Memory replay gans: Learning to generate new categories without forgetting," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[38] F. Zhu, X.-Y. Zhang, C. Wang, F. Yin, and C.-L. Liu, "Prototype augmentation and self-supervision for incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5871–5880.

[39] L. Pellegrini, G. Graffieti, V. Lomonaco, and D. Maltoni, "Latent replay for real-time continual learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 203–10 209.

[40] J. Kong, Z. Zong, T. Zhou, and H. Shao, "Condensed prototype replay for class incremental learning," *arXiv preprint arXiv:2305.16143*, 2023.

[41] T. L. Hayes, K. Kafle, R. Shrestha, M. Acharya, and C. Kanan, "Remind your neural network to prevent catastrophic forgetting," in *European Conference on Computer Vision*. Springer, 2020, pp. 466–483.

[42] M. Y. Harun, J. Gallardo, T. L. Hayes, R. Kemker, and C. Kanan, "Siesta: Efficient online continual learning with sleep," *arXiv preprint arXiv:2303.10725*, 2023.

[43] A. Gepperth and B. Pfülb, "Image modeling with deep convolutional gaussian mixture models," *arXiv preprint arXiv:2104.12686*, 2021.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[45] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[46] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2921–2926.

[47] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and . y. Ng, "Reading digits in natural images with unsupervised feature learning."

[48] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[49] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[50] M. Mundt, S. Lang, Q. Delfosse, and K. Kersting, "Cleva-compass: A continual learning evaluation assessment compass to promote research transparency and comparability," *arXiv preprint arXiv:2110.03331*, 2021.

[51] T. Lesort, A. Gepperth, A. Stoian, and D. Filliat, "Marginal replay vs conditional replay for continual learning," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 466–480.

[52] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, "Learning to learn without forgetting by maximizing transfer and minimizing interference," *arXiv preprint arXiv:1810.11910*, 2018.

[53] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18 661–18 673, 2020.

[54] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *Advances in neural information processing systems*, vol. 33, pp. 9912–9924, 2020.

[55] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[56] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. 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*, 2021, pp. 9588–9597.

[57] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," *Advances in neural information processing systems*, vol. 29, 2016.

[58] A. Gepperth, "A new perspective on probabilistic image modeling," in *International Joint Conference on Neural Networks(IJCNN)*, 2022.

[59] O. Ostapenko, T. Lesort, P. Rodriguez, M. R. Arefin, A. Douillard, I. Rish, and L. Charlin, "Continual learning with foundation models: An empirical study of latent replay," in *Conference on Lifelong Learning Agents*. PMLR, 2022, pp. 60–91.

[60] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22 243–22 255, 2020.

[61] J. Gallardo, T. L. Hayes, and C. Kanan, "Self-supervised training enhances online continual learning," *arXiv preprint arXiv:2103.14010*, 2021.