Compressed Latent Replays for Lightweight Continual Learning on Spiking Neural Networks

Compressed Latent Replays for Lightweight Continual Learning on Spiking Neural Networks

Alberto Dequino[1,2], Alessio Carpegna[2], Davide Nadalini[1,2], Alessandro Savino[2], Luca Benini[1,3], Stefano Di Carlo[2], Francesco Conti[1] ([1]Universitá di Bologna, [2]Politecnico di Torino, [3]ETH Zurich)

***Abstract*—Rehearsal-based Continual Learning (CL) has been intensely investigated in Deep Neural Networks (DNNs). However, its application in Spiking Neural Networks (SNNs) has not been explored in depth. In this paper we introduce the first memory-efficient implementation of Latent Replay (LR)-based CL for SNNs, designed to seamlessly integrate with resource-constrained devices. LRs combine new samples with latent representations of previously learned data, to mitigate forgetting. Experiments on the Heidelberg SHD dataset with Sample and Class-Incremental tasks reach a Top-1 accuracy of 92.5% and 92%, respectively, without forgetting the previously learned information. Furthermore, we minimize the LRs' requirements by applying a time-domain compression, reducing by two orders of magnitude their memory requirement, with respect to a naïve rehearsal setup, with a maximum accuracy drop of 4%. On a Multi-Class-Incremental task, our SNN learns 10 new classes from an initial set of 10, reaching a Top-1 accuracy of 78.4% on the full test set [1].**

***Index Terms*—Continual Learning, Latent Replay, Spiking Neural Networks, Edge Computing, Time Compression**

I. Introduction

The current landscape of edge AI is dominated by Artificial Neural Networks (ANNs) driven by high-end server models, like Transformers and Convolutional Neural Networks (CNNs) [1]. Concurrently, the recent advancements in hardware and software for edge Internet of Things (IoT) devices have enabled the integration of Artificial Intelligence (AI) on low-power sensor nodes. However, deploying complex ANNs for edge inference involves imposing constraints like quantization and pruning [14] to accommodate small IoT devices, like ultra-low-power microcontrollers. Moreover, edge AI models are susceptible to errors, once deployed, due to shifts in data distribution between training and operational environments [2]. Also, an increasing number of applications require adapting AI algorithms to individual users while maintaining privacy and minimizing internet connectivity.

Continual Learning (CL) — i.e., the ability to continually learn from evolving environments, without forgetting previously acquired knowledge — emerges as a novel paradigm to address these challenges. *Rehearsal-based* methods [17], [20], the most accurate CL solutions, mitigate forgetting by continually training the learner on a mix of new data and a stored set of samples from previously learned tasks, albeit at the expense of additional on-device storage. *Rehearsal-free* methods [5], [15] rely on tailored modifications to network architecture or learning strategy to ensure model resilience to forgetting, without saving samples on-device, but with a potential trade-off in accuracy. CL at the edge, especially Rehearsal-based methods, can be resource-intensive for various ANN models, as CNNs, demanding substantial on-device storage for complex learning data.

In this context, Spiking Neural Networks (SNNs) emerge as a promising energy-efficient paradigm, exhibiting high accuracy and efficiency in processing time series [11]. SNNs closely emulate biological neurons' behavior, communicating with spikes that can be efficiently stored as 1-bit data in digital architectures. This simplified data encoding creates new opportunities for developing CL solutions. While online learning has been explored in both hardware and software SNNs implementations [7], CL strategies in SNNs are only partially investigated in Rehearsal-free methods [12], [19], [21]. To the best of our knowledge, only Proietti, et al. [18] addressed Rehearsal-based CL in SNNs. However, their work shows limited accuracy and does not optimize memory storage, which is vital for edge devices. This work makes the following contributions:

- A cutting-edge, memory efficient implementation of Rehearsal-based CL for SNNs, designed to seamlessly integrate with resource-constrained devices. We enable CL on SNNs by exploiting a Rehearsal-based algorithm — i.e., *Latent Replay (LR)* — proving to achieve State-of-the-Art classification accuracy on CNNs [17].
- A novel approach to reduce the rehearsal memory, based on the robustness of information encoding in SNNs to precision reduction, which allows us to apply a lossy compression on the time axis.
- We validate our method targeting a keyword spotting application, using Recurrent SNN, on two common CL setups: *Sample-Incremental* and *Class-Incremental* CL.
- Finally, to highlight the proposed approach's robustness, we test it in an extensive Multi-Class-Incremental CL routine, learning 10 new classes from an initial set of 10 pre-learned ones.

In the *Sample-Incremental* setup, we achieve a Top-1 accuracy of 92.46% on the Spiking Heidelberg Dataset (SHD) test set, requiring 6.4 MB for the LRs. This occurs while incorporating a new scenario, for which the accuracy is improved by 23.64%, without forgetting the previously learned ones. In the Class-Incremental setup, a Top-1 accuracy of 92% was attained, requiring 3.2 MB, when learning a new class with an accuracy of 92.50%, accompanied by a loss of up to 3.5% on the old classes. When jointly applying compression

[1]To encourage research in this field, we release the code related to our experiments as open-source: https://github.com/Dequino/Spiking-Compressed-Continual-Learning

and selecting the most performing LR index, we reduce by 140× the memory requirements for the rehearsal data, while losing only up to 4% accuracy, with respect to the naïve approach. Moreover, in the Multi-Class-Incremental setup, we achieve 78.4% accuracy with compressed rehearsal data, while learning the set of 10 new keywords. These results pave the way to a new low-power and high-accuracy approach for CL on edge.

II. Related Work

CL on ANNs involves two main approaches: Rehearsal-based and Rehearsal-free methods. In Rehearsal-based methods, mitigating catastrophic forgetting involves training the learner on a mix of newly acquired data and samples from previously learned tasks. To increase learned classes, iCaRL [20] utilizes a set of representative old class examples as rehearsal data, chosen to maintain a balanced set of classes. To improve the efficiency of Rehearsal-based methods, Pellegrini et al. [17] propose storing rehearsal data as Latent Replays, i.e., activations produced as the output of one of the hidden layers of the learner, specifically a feed-forward CNN. Only the last layers are retrained, while the earliest backbone's weights are frozen. Results, compared to iCaRL, demonstrate three orders of magnitude faster execution, with an almost 70% improvement in Top-1 accuracy on a Class-Incremental setup using the CORe50 dataset. In Rehearsal-free methods, addressing forgetting involves modifying the learner's architecture or customizing the training procedure [5], [15]. However, their accuracy lags behind Rehearsal-based approaches. We adopt the Rehearsal-based approach using LRs, given its demonstrated effectiveness on CNNs.

Continual Learning in SNNs has primarily focused on Rehearsal-free methods. Skatchkovsky et al. [21] propose a Bayesian continual learning framework, providing well-calibrated uncertainty quantification estimates for new data. In contrast, the SpikeDyn framework [19] introduces unsupervised continual learning on a model search algorithm, supporting adaptive hyperparameters. However, this approach performs poorly, achieving 90% accuracy on average for a new class in the MNIST dataset while only maintaining 55% accuracy on the old classes. Drawing inspiration from human brain neurons, Han et al. [12] propose Self-Organized Regulation networks, capable of learning new tasks by self-organizing the neural connections of a fixed architecture. Additionally, they simulate irreversible damage to SNNs structures by pruning the models. Results on the CIFAR100 and Mini-ImageNet datasets, using DNN-inspired Convolutional and Recurrent networks, demonstrate an average accuracy of around 80% and 60%, respectively, for all learned classes. To the best of our knowledge, only the work of Proietti, et al., [18] targets Rehearsal-based CL on SNNs. In their approach, a Convolutional SNN model is trained to learn in a Class-Incremental and Task-free manner on the MNIST dataset, learning multiple binary classification tasks in sequence. However, their exploration doesn't involve any memory optimization technique, storing raw data for the rehearsal phase. Additionally, they report a top-1 accuracy of 51% after learning sequentially the 10 target classes, while requiring 16MB of rehearsal memory.

To improve accuracy and optimize memory efficiency, we explore the reharsal-based domain, applying efficient time-compressed LRs to SNNs.

III. Background

SNNs are a category of neural networks inspired by the behavior of biological brain segments. Unlike other ANNs models, SNNs convey information through sequences of spikes, with their representation adapting to the execution domain. In the digital domain, considered in this study, spikes are encoded as binary values (one for a spike, zero otherwise) associated with discrete timesteps. The information conveyed by spikes is encoded in their sequence, such as their instantaneous rate, arrival time, or more complex patterns. Following Eshraghian, et al., [6] and aiming at the classification of signals with dense time-domain information (e.g., sound samples), we use a Fully Connected Recurrent SNN architecture with L layers, incorporating intra-layer side connections akin to those found in [4] and [23]. The chosen model employs a Synaptic Conductance-based Second Order Leaky Integrate and Fire (LIF) neuron model [3]. Fig. 1-a) illustrates the structure of the recurrent neurons, showcasing a collection of feed-forward weights denoted as W linked to each input and a set of recurrent weights labeled as V, each one connected to the output of the neurons within the same layer, including itself. Optimally training SNN models poses a challenge, due to the non-differentiability of the spike function. Similar to Recurrent Neural Networks (RNNs), state-of-the-art back-propagation techniques can be applied to SNNs using Surrogate Gradients (SGs) [16], implemented through Back-Propagation Through Time (BPTT) [22]. In this study, SNNs undergo training via BPTT, utilizing a fast-sigmoid SG to approximate the step function characteristic of Recurrent Neurons.

In this work, we target CL, a machine learning paradigm that enables models to continuously acquire knowledge from a data stream without erasing prior learning. Let us consider a dataset D on which the model is pre-trained on $K_{class} < N_{class}$ classes and/or $K_{scene} < N_{scene}$ potential scenarios. A scenario here represents a specific data subset containing all N_{class} classes, exemplifying a distinct data variation (e.g., a single speaker in a keyword-spotting dataset). Let T be a test set comprising samples from N_{class} classes and N_{scene} scenarios. In the CL paradigm, the pre-trained model continually incorporates and learns new labeled samples $\notin D$, enhancing its predictive accuracy on T. CL algorithms require specific techniques to prevent models from forgetting previously learned data, a challenge known as *Catastrophic Forgetting* [13], which has recently been demonstrated to exist also in SNNs [9], [10].

IV. Latent Replay-based Continual Learning in SNNs

In this paper, we draw inspiration from the work of Pellegrini et al. [17] on LR, initially designed for CNNs. Several crucial steps must be addressed to adapt this paradigm for SNNs.

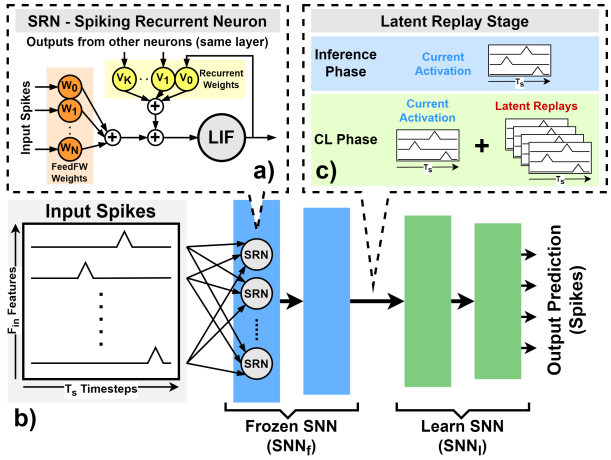

Fig. 1: Visual depiction of a generic SNN with Recurrent Neurons and Latent Replays. In a), our Recurrent Neuron model; in b), the structure of a generic Fully-Connected model for Continual Learning; in c), an example of the data fed to the Latent Replay stage.

Algorithm 1 SNN Latent Replay training

1: **Input:** TR_{pre}, TR_{cl}.
2: **Parameters:** E_{pre}, E_{cl}, K, η.
3:
4: **Initial Training phase:**
5: Initialize SNN weights W, V randomly
6: **for** $e \leftarrow 1$ to E_{pre} **do**
7: $(W, V) \leftarrow$ BPTT.train(SNN,TR_{pre},(W, V), η)
8: **end for**
9:
10: **Prepare network for CL:**
11: $(\mathrm{SNN_f}, \mathrm{SNN_l})$ = split (SNN, K)
12: LR = Inference($TR_{replay} \subseteq TR_{pre}$, $\mathrm{SNN_f}$)
13:
14: **Train network on new data:**
15: **for** $e \leftarrow 1$ to E_{cl} **do**
16: A = Inference(TR_{cl}, $\mathrm{SNN_f}$)
17: (W_l, V_l) = BPTT.train ($\mathrm{SNN_l}$, $A \cup LR$, (W_l, V_l), η)
18: **end for**

First, rehearsal data need to encapsulate the temporal evolution of a layer of spiking neurons, in contrast to the static input images in the original LR framework. Second, this temporal evolution must fit into the learning algorithm, i.e., BPTT, while mixing with newly acquired data. Furthermore, our attention is directed towards two CL scenarios: a Sample-Incremental setup, where a learner is trained to classify unseen scenarios, and a Class-Incremental setup, where the learner is tasked with identifying an increasing number of classes.

A. Latent Replays in SNNs

Algorithm 1 outlines the proposed Latent Replay-based training for an SNN with L layers. The neural network undergoes pre-training over E_{pre} epochs using BPTT on an initial training set TR_{pre} with an η learning rate (lines 4-8). As illustrated in Fig. 1-b), the network is then split into two sections: $(\mathrm{SNN_f})$, comprising the first K layers denoted as *frozen layers*, and $(\mathrm{SNN_l})$, encompassing layers from the $L-K+1$-th to the L-th, marked as *learning layers* (line 11). The latent replays (Fig. 1-c), denoted as LR, constitute a collection of the output activations of the K^{th} layer when exposed to a subset TR_{replay} of the pre-training set. This collection must be stored for later use during CL training (line 12), necessitating onboard storage. Since input data are trains of spikes (i.e., binary values) distributed over T_s time-steps, the stored LRs are in turn sequences of T_s single-bit activations.

When training the network on new data, belonging to a new scenario or class, only the *learning layers*, are trained: the *frozen layers* simply propagate the input spikes sequences in the forward direction, processing them through the function learned in the pre-training phase (line 16). The *learning layers* are then trained for E_{cl} epochs, using the output activations of the K^{th} layer, blended with the stored LRs to keep memory of the learned data (line 17).

B. Optimization of LR memory

To keep memory of previously learned information, an additional storage is required for the LRs. The problem becomes increasingly serious when more classes or scenarios are learned sequentially, making the required memory grow over time. To limit the phenomenon, we introduce a time-domain lossy compression on LRs.

Fig. 2 illustrates the compression technique applied to a generic spiking sequence. Given a compression ratio (C_r), each component of a LRs sample is compressed into a sequence of T_s/C_r binary activations. The algorithm involves two steps: first, the uncompressed sequence is divided into chunks composed of C_r activations. Subsequently, each chunk is compressed in a single time step using a lossy compression based on thresholding: if the number of spikes within the chunk reaches a given *threshold* (e.g., 1), a spike is generated in the compressed sequence. The impact on computational resources manifests as a $C_r\times$ memory reduction fo the LRs.

When processing a LR with the compressed data, decompression is needed to respect the time scale of the model. To avoid this step, we attempted to change the model's time constants, adapting them to the compressed duration of the spike sequences. However, the accuracy loss was dramatic, especially with compressed LR performed in the first layers. We hypothesize that the inter-layer recurrent connections in these layers were tuned to a specific time scale and did not adapt well to its variation. In support of this idea, we observed that a compressed LR on the last layer, which doesn't contain any feedback connection, almost doesn't affect the final accuracy. The solution, which also works for recurrent layers, is to uncompress the sequence by interleaving the compressed samples with zeroes to match the time scale of the spikes' sequence at run-time.

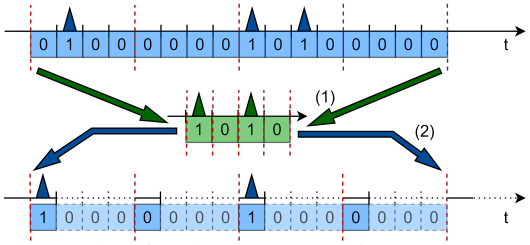

Fig. 2: Example of a lossy compression (1) to store LRs. Compression ratio is here set to 4:1. Shrinking LRs and activations reduces the memory by 4×. A de-compression step (2) is required to respect the SNN time scale.

V. Experimental Results

A. Experimental Setup

We evaluated our methodology using the SHDs dataset [4], which comprises 10,420 spiking trains of 100 timesteps each. These trains consist of audio samples obtained through a conversion system inspired by the human cochlea. The dataset is categorized into 20 classes, corresponding to numbers 0 to 9, pronounced by 12 speakers in both English and German. We employed a recurrent SNN with 700 input neurons and 20 outputs (one for each class), including 4 layers with decreasing output size (200-100-50-20). As described in Section III, we adopted a Synaptic Conductance-based Second Order LIF neuron model, trained using BPTT and a fast-sigmoid SG. Our training setup incorporates a learning rate of $\eta = 10^{-3}$ for the Sample Incremental task and $\eta = 2 \cdot 10^{-4}$ for the Class Incremental task.

B. Weights initialization

When pre-training an SNNs for a Class-Incremental setup, neurons associated with unlearned classes are trained to be inactive. Therefore, when adding a new class to the pre-trained classifier, the yielded accuracy is poor, reaching a maximum of 57%. This issue can be addressed by re-initializing the neuron weights devoted to detect the new class. When performing weight re-initialization, the obtained accuracy is strictly linked to the re-initialization strategy. Notably, random, constant, and Xavier-Glorot initializations [8] proved inefficient for proficient learning of the new class. Instead, we adopted a normal random distribution with mean and variance aligned with those of the classifier's weights trained on the old classes. This approach enables our model to achieve a remarkable 92.9% accuracy on the new class. Further details are discussed in Section V-D. Unlike the Class-Incremental scenario, the Sample-Incremental scenario does not require any initialization, as the number of classes does not change between samples.

C. Sample-Incremental CL

Our model undergoes pre-training on 11 out of 12 scenarios (speakers) to simulate user personalization in keyword spotting tasks. The 12^{th} scenario is introduced using a Sample-Incremental CL approach. To benchmark our technique against methodologies like [18], we conduct experiments in three setups: (i) a *naïve incremental* setup, where the 12^{th} speaker is learned without rehearsal; (ii) a *naïve rehearsal* setup, mixing the 12^{th} speaker's input activations with 2,560 samples from the training step; (iii) the proposed LR setup, integrating 2,560 LRs stored at the output of the 2^{nd}-to-last layer. Fig. 3-a) displays the forgetting measurement, represented by the Top-1 accuracy of the SNN after 50 CL epochs, focusing on the old scenarios. Fig. 3-b) showcases the accuracy of the new scenario.

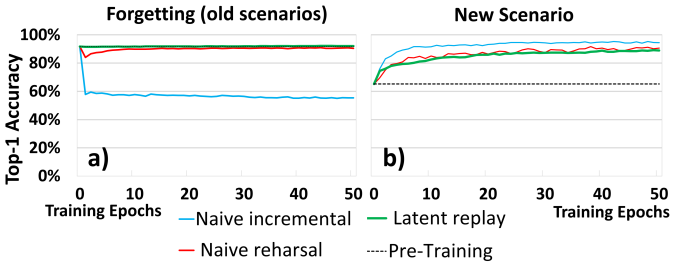

Fig. 3: Measurement of (a) forgetting and (b) Top-1 accuracy on the new scenario (Sample-Incremental CL). Latent Replays are applied to the 2^{nd} to last layer of our model using 2,560 past samples for the replay.

As observed in CNNs, SNNs face Catastrophic Forgetting when learning new samples without rehearsal. In naïve incremental, few epochs specialize the model toward the new speaker ($> 90\%$ accuracy), resulting in almost 40% accuracy loss on the old speakers. In naïve rehearsal, rehearsal data enable the model to retain nearly the same accuracy before the CL routine, limiting the accuracy drop on the old speakers to a 2%. Conversely, the new speaker is learned with almost 90% accuracy. Retraining only the last 2 layers with LRs maximizes accuracy retention on old scenarios. The pre-training accuracy of 11 speakers is enhanced by 1%, as the new speaker's samples strengthen classification in classes seen during pre-training. The Top-1 accuracy on the new speaker is increased by 24%, reaching an overall 88%. This is 2% lower than naïve rehearsal, but it is associated with a 7× reduction in the memory required to store rehearsal data.

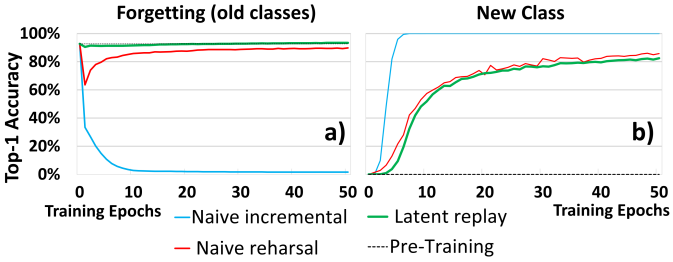

Fig. 4: Measurement of (a) forgetting and (b) Top-1 accuracy on the new class (Class-Incremental CL). Latent Replays are applied to the 2^{nd} to the last layer of our model.

D. Class-Incremental CL

In this scenario, we simulate the introduction of a new keyword by pre-training our model on 19 out of 20 classes and adding the 20^{th} class subsequently. Similar to the Sample-Incremental setup, we compare (i) a *naïve incremental*, (ii) a *naïve rehearsal*, and (iii) *the proposed LR setup*. Fig. 4-a) measures forgetting, while 4-b) shows Top-1 accuracy on the new class. In terms of forgetting, LRs show no accuracy

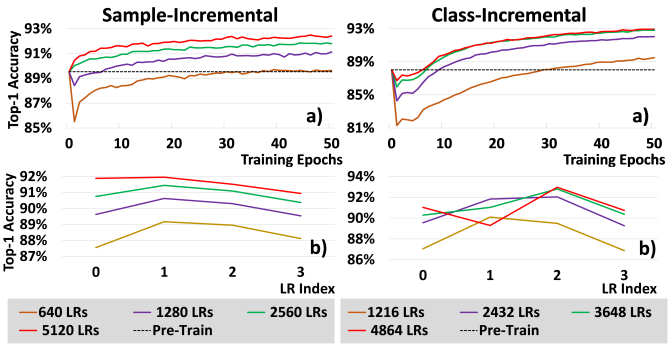

Fig. 5: Top-1 accuracy on SHD's test set in Sample and Class-Incremental CL, in case of a variable number of LRs (a) with respect to the epochs, (b) with respect to the LR index.

drop, outperforming both naïve rehearsal ($\simeq$ 3% drop) and naïve incremental (complete forgetting). The new class is learned within 50 epochs for all three methods. While naïve incremental overfits the new class, completely forgetting the past ones, LRs achieve the highest accuracy of 83% without notable forgetting. Again, the accuracy reached by naïve rehearsal is a bit higher on the new data, and exceeds LRs by 3%. However this comes at the cost of a 3% forgetting of the past classes, with a 7× memory occupation.

E. Classification Accuracy vs Number of LRs

Fig. 5 comprehensively analyzes the impact of various hyperparameters on accuracy, examining (a) the number of LRs and (b) LR indices across different epochs. In the Sample-Incremental setup, increasing LRs positively influences Top-1 accuracy. The primary effect is on the maximum accuracy, rather than the convergence slope, which remains relatively constant across all cases. An initial forgetting can be observed with 640 and 1280 past samples, becoming more and more accentuated with decreasing numbers of LRs. However, in all the cases the network is able to retrieve the past information, bringing the accuracy back to its initial value, or exceeding it. The choice of LRs index impacts maximum accuracy as well: figure 5 shows the superiority of LRs (indexes 1, 2 and 3) with respect to naïve reharsal (index 0). The highest accuracy of 92.5% is achieved with 5,120 LRs at layer index 1, as layer 0 retains knowledge of old data while extending towards old and new samples.

Increasing LRs has similar effects for the Class-Incremental CL setup. While varying the layer index, we observe a peak accuracy at LRs index 2 , making Top-1 accuracy reach 93% with 4,864 LRs. This indicates the capability of the final layers to learn the new class, exploiting the higher-level features learned by previous layers. An interesting effect can be noted at index 1, where the curve is non-monotonic: in this case, a larger number of LRs delays the convergence of the model, preventing it from reaching an acceptable accuracy within the 50 epochs.

F. Compressed LRs

We now explore the memory-accuracy trade-off of our LR compression algorithm. The choice of the LRs' layer index and the time compression factor C_r is based on the trade-off between accuracy and memory requirements. Tables I and II present the results obtained on the full SHD test set, for Sample and Class-Incremental setups respectively. In the case of *naïve rehearsal*, the rehearsal data is stored as past input activations, consisting of 100 timesteps and 700 inputs. The spatial size of LRs aligns with the size of the layer on which they are stored as input, such as 100 inputs for layer 2. After selecting a C_r, the rehearsal data is additionally compressed on the time axis; for example, with $C_r = 5$, the data is reduced by 5×, featuring 20 timesteps. Therefore, all the data is scaled proportionally. Our experiments show a minimal size of the rehearsal data, with LRs collected on layer 3 using $C_r = 10$, being 140× smaller than *naïve rehearsal* without compression. For the Sample-Incremental setup (Tab. I), we observe that a Top-1 accuracy of 92.46% is reached for uncompressed LRs with index 1, surpassing the naïve rehearsal by 2%. Compressed LRs reduce the maximum accuracy: in case of a 1:5 ratio, 90.27% is achieved with index 3, while the accuracy drop on earlier LR indices becomes prohibitive before layer 2. This drop is accentuated for 1:10 compression, achieving a Top-1 accuracy of 88.79% on index 3. However, a memory save of 10× is attested.

TABLE I: Memory-Accuracy tradeoff for Sample-Incremental CL, with 2560 LRs and variable C_r and LR index. Pre-training accuracy on the scenario to be learned was 65%.

C_r \ LRs		Naïve rehearsal	Layer 1	Layer 2	Layer 3
1:1	**Acc**	90.50%	**92.46%**	91.79%	90.37%
	Mem	22.4 MB	**6.4 MB**	3.2 MB	1.6 MB
1:5	**Acc**	67.81%	79.71%	89.91%	**90.27%**
	Mem	4.48 MB	1.28 MB	640 KB	**320 KB**
1:10	**Acc**	67.12%	68.12%	84.73%	**88.79%**
	Mem	2.24 MB	640 KB	320 kB	**160 kB**

TABLE II: Memory-Accuracy tradeoff for Class-Incremental CL, with 2432 LRs and variable C_r and LR index.

C_r \ LRs		Naïve rehearsal	Layer 1	Layer 2	Layer 3
1:1	**Acc**	89.55%	91.83%	**92.05%**	89.26%
	Mem	22.4 MB	6.4 MB	**3.2 MB**	1.6 MB
1:5	**Acc**	84.93%	57.43%	82.03%	**86.78%**
	Mem	4.48 MB	1.28 MB	640 KB	**320 KB**
1:10	**Acc**	77.43%	30.75%	69.42%	**85.53%**
	Mem	2.24 MB	640 KB	320 kB	**160 kB**

Also in case of a Class-Incremental setup (Tab. II), the Top-1 accuracy of Latent-Replay, obtained at LR index 2, surpasses the Naive-Incremental by a 2.5%. Performing a LR at index 3 brings back the accuracy to 89.26%, comparable with Naïve reharsal, but with a memory requirement 14× lower. Finally 1:5 and 1:10 time compressions correspond to a further drop of 2.5% and 3.7%, but with a memory saving of 5× and 10× . The accuracy loss for previous indexes is still too high, indicating that a lighter compression is required.

G. Comparison with other compressions

Our technique focuses on preserving most of the temporal content of spike sequences, while allowing for an easy decompression at run-time. Alternatively, latent spikes can be aggregated by storing them as active spike counts.

We test two additional compression methods, which we

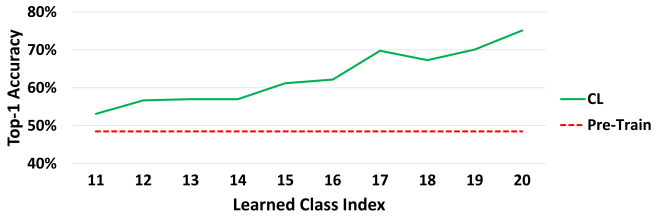

Fig. 6: Multi-Class-Incremental CL on SHD. Here, a pre-trained model learns to classify 10 more classes, starting from a pre-training on 10 classes.

compare with our sub-sampling approach (Fig.2): first, we aggregate spikes over the full sequence and expand during rehearsal, placing all spikes in the first time-steps. While this drastically reduces the required memory to $log_2(N)$ bits/sequence, the time information is lost, causing a 12% to 14% accuracy drop compared to our technique, with C_r = 10 and 20. The second is a hybrid approach: we divide sequences into chunks and accumulate spikes within each chunk. This leads to a 2% accuracy improvement with respect to our method, at the cost of a $3\times$ and $4\times$ memory occupation with $C_r = 20$ and $C_r = 10$, respectively. Therefore, for the considered keyword spotting setup, our method provides an optimal trade-off between accuracy, memory occupation and run-time decompression complexity. Also, the temporal properties of the spike trains prove to be more relevant than the absolute number of spikes.

H. Multi-Class-Incremental Setup

We now provide a more complex testing scenario, i.e., a keyword spotting language shift in which our model learns to classify 10 classes of digits in German, starting from 10 pre-learned English-spoken digits. The pre-trained knowledge is stored as 2560 LRs, 256 per-class. After each new class is learned, 256 LRs are added to the memory. Our model is trained for 50 epochs per class, with $C_r = 2$.

Fig. 6 shows the Top-1 accuracy over the 20-class SHD test set. The pre-training accuracy is of 48.5%, corresponding to an accuracy of 97% on the first 10 classes only. While learning with CL, a monotonic growth is assessed, learning each successive class with 88.2% average accuracy. For each new class learned, we observe an average forgetting of 2.2% on all the previous classes. In the end, we observe a final accuracy of 78.4% on the full test set.

REFERENCES

[1] Ayoub Benali Amjoud and Mustapha Amrouch. Object detection using deep learning, cnns and vision transformers: A review. *IEEE Access*, 11:35479–35516, 2023.

[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[3] Romain Brette and Wulfram Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94(5):3637–3642, 2005.

[4] Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, and Friedemann Zenke. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):2744–2757, 2022.

[5] Benjamin Ehret, Christian Henning, Maria R. Cervera, Alexander Meulemans, Johannes von Oswald, and Benjamin F. Grewe. Continual learning in recurrent neural networks, 2021.

[6] Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

[7] Charlotte Frenkel and Giacomo Indiveri. Reckon: A 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 65, pages 1–3, 2022.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[9] Ryan Golden, Jean Erik Delanois, Pavel Sanda, and Maxim Bazhenov. Sleep prevents catastrophic forgetting in spiking neural networks by forming joint synaptic weight representations. *bioRxiv*, 2020.

[10] Ilyass Hammouamri, Timothée Masquelier, and Dennis Wilson. Mitigating Catastrophic Forgetting in Spiking Neural Networks through Threshold Modulation. *Transactions on Machine Learning Research Journal*, November 2022.

[11] Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 388–404, Cham, 2020. Springer International Publishing.

[12] Bing Han, Feifei Zhao, Wenxuan Pan, Zhaoya Zhao, Xianqi Li, Qingqun Kong, and Yi Zeng. Adaptive reorganization of neural pathways for continual learning with spiking neural networks, 2023.

[13] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[14] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[15] Vincenzo Lomonaco, Davide Maltoni, Lorenzo Pellegrini, et al. Rehearsal-free continual learning over small non-iid batches. In *CVPR Workshops*, volume 1, page 3, 2020.

[16] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[17] 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, 2020.

[18] Michela Proietti, Alessio Ragno, Roberto Capobianco, et al. Memory replay for continual learning with spiking neural networks. In *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2023.

[19] Rachmad Vidya Wicaksana Putra and Muhammad Shafique. Spikedyn: A framework for energy-efficient spiking neural networks with continual and unsupervised learning capabilities in dynamic environments. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1057–1062, 2021.

[20] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[21] Nicolas Skatchkovsky, Hyeryung Jang, and Osvaldo Simeone. Bayesian continual learning via spiking neural networks. *Frontiers in Computational Neuroscience*, 16, 2022.

[22] P.J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[23] Bojian Yin, Federico Corradi, and Sander M. Bohte. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks, Mar 12 2021.