Class-incremental Learning for Time Series: Benchmark and Evaluation

Zhongzheng Qiao
qiao0020@e.ntu.edu.sg
IGP-ERI@N, NTU
I^{2}R, A*STAR
CNRS@CREATE
Singapore

Quang Pham
Zhen Cao
pham_hong_quang@i2r.a-star.edu.sg
Cao_Zhen@i2r.a-star.edu.sg
I^{2}R, A*STAR
Singapore

Hoang H. Le
hoangle7910@gmail.com
Ho Chi Minh University of Science,
Vietnam National University
Ho Chi Minh City, Vietnam

P. N. Suganthan
p.n.suganthan@qu.edu.qa
Qatar University
Doha, Qatar

Xudong Jiang
exdjiang@ntu.edu.sg
School of Electrical and Electronic
Engineering, NTU
Singapore

Savitha Ramasamy
ramasamysa@i2r.a-star.edu.sg
I^{2}R, A*STAR
CNRS@CREATE
Singapore

ABSTRACT

Real-world environments are inherently non-stationary, frequently introducing new classes over time. This is especially common in time series classification, such as the emergence of new disease classification in healthcare or the addition of new activities in human activity recognition. In such cases, a learning system is required to assimilate novel classes effectively while avoiding catastrophic forgetting of the old ones, which gives rise to the Class-incremental Learning (CIL) problem. However, despite the encouraging progress in the image and language domains, CIL for time series data remains relatively understudied. Existing studies suffer from inconsistent experimental designs, necessitating a comprehensive evaluation and benchmarking of methods across a wide range of datasets. To this end, we first present an overview of the Time Series Class-incremental Learning (TSCIL) problem, highlight its unique challenges, and cover the advanced methodologies. Further, based on standardized settings, we develop a unified experimental framework that supports the rapid development of new algorithms, easy integration of new datasets, and standardization of the evaluation process. Using this framework, we conduct a comprehensive evaluation of various generic and time-series-specific CIL methods in both standard and privacy-sensitive scenarios. Our extensive experiments not only provide a standard baseline to support future research but also shed light on the impact of various design factors such as normalization layers or memory budget thresholds. Codes are available at https://github.com/zqiao11/TSCIL.

CCS CONCEPTS

• **Computing methodologies** → **Lifelong machine learning**; **Neural networks**.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08
https://doi.org/10.1145/3637528.3671581

KEYWORDS

Class-incremental Learning, Continual Learning, Time Series Classification

ACM Reference Format:
Zhongzheng Qiao, Quang Pham, Zhen Cao, Hoang H. Le, P. N. Suganthan, Xudong Jiang, and Savitha Ramasamy. 2024. Class-incremental Learning for Time Series: Benchmark and Evaluation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671581

1 INTRODUCTION

Time series (TS) data play a pivotal role in various domains such as acoustics, healthcare, and manufacturing [55]. Typical deep learning approaches for time series classification [29] are trained on a static offline dataset, collected prior to training and under the assumption that the data are independent and identically distributed (i.i.d.). However, real-world applications often challenge this i.i.d. assumption, as practical systems usually operate in dynamic environments with non-stationary data streams where the underlying data distributions keep evolving. For instance, a TS-classification model for human activity recognition or gesture recognition should be capable of adapting to newly introduced classes [11, 47]. In such scenarios, the challenge of developing an adaptive learner lies not only in seamlessly assimilating new concepts from incoming data but also in simultaneously preserving and accumulating knowledge of all encountered classes.

The primary challenge in this endeavor stems from the well-known *stability-plasticity* dilemma [23], where the model must be *stable* enough to remember its past knowledge, while being *plastic* to accommodate new information. However, current findings [34, 43] suggest that neural networks are too plastic as they cannot retain old knowledge while learning the newer ones, which is referred to as the *catastrophic forgetting* phenomenon [50]. Thus, developing efficient methodologies to achieve a good trade-off between facilitating learning new skills and alleviating catastrophic forgetting has played a central role in the development of continual learning (CL). Extensive efforts have been devoted to exploring various continual learning scenarios [40], and Class-incremental

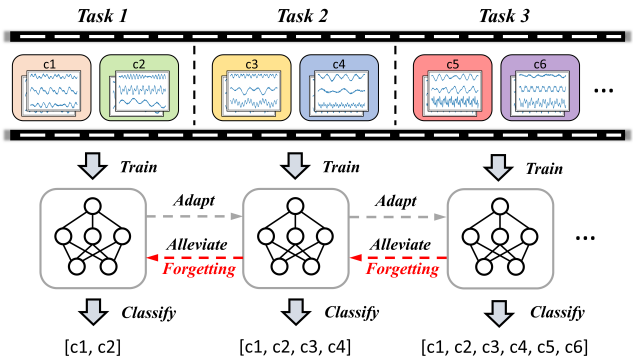


Figure 1: Schematic of Time Series Class-incremental Learning (TSCIL) process on a dynamic task sequence. Each task introduces new classes (c1 to c6), separated by clear task boundaries. The model undergoes sequential training on the tasks. After training on each task, the model needs to recognize all classes encountered thus far without catastrophic forgetting. The previously learned parameters are adapted for next task's learning.

Learning (CIL) [57, 73] emerges as the most prominent and challenging one. Nevertheless, the majority of such studies only explore image [49] or language [33] applications. On the other hand, time series, despite its ubiquity and continuous nature, remains under investigated by the community. Existing studies suffer from inconsistency in various aspects of experimental setups, including datasets [21, 35], normalization [11, 54], and learning protocol [47, 67], etc.

To bridge this gap, this paper serves as a pioneering effort focusing exclusively on Class-incremental Learning for time series data (TSCIL). We first provide an overview of TSCIL, including problem definition, specific challenges and related works. We focus on investigating the unique characteristics of TS data such as data privacy and intra-class variations and their impact on CIL. The key contribution is the development and open-sourcing of a benchmark to facilitate a standardized evaluation of both generic and TS-specific CIL methods across various real-world datasets. This framework represents a useful resource for the research community, offering an adaptable code base for easy integration of new datasets, algorithms, and tailored learning setups, thereby empowering researchers to further develop the field of TSCIL.

Our experiments commence with the standard academic setup, evaluating both generic and TS-specific CIL methods based on regularization and experience replay [16, 72]. We further investigate the influence of different factors to the CIL performance, including normalization, memory budget and classifier type. Besides the standard setting, we also consider two application-specific scenarios particularly pertinent to TS modality. First, we investigate the privacy-sensitive environments, where the TS data are closely tied to individual users and unallowable to store the historical samples of previous tasks. Thus, we explore the generative replay strategy [64] and investigate its performance in this challenging setting. Secondly, we consider the impact of intra-class variations on TSCIL. In most datasets, time series are collected from various subjects or sources, each exhibits a distinctive input domain. Therefore, we investigate how one can incorporate such subjective information to further improve the TSCIL results.

In summary, our contributions are threefold: (1) we present a systematic overview of TSCIL, including problem definition, challenges, and existing methods. (2) We introduce a unified evaluation framework, complete with public datasets, standard protocols, and a range of methods, to facilitate further research in the field. (3) We conduct a holistic comparison of state-of-the-art CIL methodologies in both standard academic setups and application-specific scenarios, shedding light on the promises and limitations of existing methods in the context of time series data.

2 PROBLEM DEFINITION

Class-incremental Learning (CIL) involves an agent continuously learning new classes from a dynamic data stream. Following the standard academic setting [34, 69, 70], CIL represents the data stream as a sequence of tasks $\{\mathcal{T}^1, ..., \mathcal{T}^T\}$, where tasks sequentially emerge at distinct steps. Task at step t is defined as $\mathcal{T}^t = \{\mathcal{D}^t, \mathcal{Y}^t\}$, characterized by a label space $\mathcal{Y}^t$ and training data $\mathcal{D}^t = \{(\mathbf{x}_i^t, y_i^t) \,|y_i^t \in \mathcal{Y}^t\}_{i=1}^{N_t}$, where N_t is the number of samples. We assume that each task has the same number of disjoint classes, i.e., $|\mathcal{Y}^i| = |\mathcal{Y}^j|$ and $\mathcal{Y}^i \cap \mathcal{Y}^j = \varnothing$ for $i \neq j$. We focus on this setup with non-overlapping classes since the reappearing of old classes reduces the challenge of retaining past knowledge [85].

Given the task sequence, a model $f(\mathbf{x}; \theta)$ is trained on all the tasks in an incremental manner. Formally, at task $\mathcal{T}^t$, we denote the model of interest by f_{θ_t}, which is parameterized by θ_t. The optimized parameters after learning the $t-$task is defined as θ_t^*. At task $\mathcal{T}^t$, the model with parameter θ_{t-1}^* is adapted to the new task and presented only with $\mathcal{D}^t$ for training, without access to either past or future training datasets. A memory buffer $\mathcal{M}$ with fixed budget M can be optionally used, which stores a collection of historical samples for future replay (see Appendix A.1 for details). The learning objective is to enable the model to learn the new task $\mathcal{T}^t$ effectively, while also retaining knowledge from the previous tasks $\{\mathcal{T}^1, \ldots, \mathcal{T}^{t-1}\}$. Denoting the classification loss by $\mathcal{L}_c$, the ultimate learning objective to learn the entire task sequence is formulated as:

$$\theta_T^* = \underset{\theta}{\operatorname{argmin}} \sum_{t=1}^{T} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_t} [\mathcal{L}_c(f(\mathbf{x}; \theta), y)] \tag{1}$$

We adapt this standard CIL setup to time series data, thereby defining the **Time Series Class-incremental Learning** (TSCIL) problem. In this setup (see Figure 1), each sample is a time series $\mathbf{x} \in \mathbb{R}^{C\times L}$, where C indicates the number of channels/variables, and L denotes the length of the series. TSCIL not only inherits the constraints of standard CIL but also has its own challenges, which we highlight in the following.

- **Normalization**: In image-based CIL, it is common to normalize images using the statistics computed from ImageNet [60], scaling the pixel density to the range of $[0, 1]$. However, such approaches are not directly applicable to time series due to the lack of large scale datasets encompassing many patterns.

Given that data normalization is often overlooked in time series data [11, 36, 82], we present a practical solution to this issue in Section 4.2.

- **Data Privacy**: Applications involving TS data are often associated with the need to preserve the privacy of data [20]. This necessitates methods to avoid retaining the original user data to safeguarding privacy. Using synthetic samples has shown as a viable solution to preserve user privacy [64, 81], which is particularly evaluated in Section 5.2.
- **Intra-class Variation**: Time series often exhibit greater intra-class variations than images [67]. This is primarily because real-world time series are collected from various sources or subjects, each having its own characteristics [5, 61]. This phenomenon results in a complex interaction in the continual learning, where not only new classes are introduced over time, but also a class might compose several patterns. We investigate this issue in Section 5.3.

After finishing each task, the model is evaluated on the test sets of all previously learned tasks. The model needs to classify all the classes from $\mathcal{Y}^{1:t} = \mathcal{Y}^1 \cup \ldots \mathcal{Y}^t$, without being provided with a task identifier. The performance of the model is evaluated with metrics introduced in Section 4.4.

3 RELATED WORKS

In existing TSCIL literature, a prevalent topic is the application of established generic CIL methods in time series scenarios. An online user authorization framework based on EWC [34] and iCaRL [57] is proposed in [11], continuously recognizing new users based on biomedical TS signals. [36] applies classic regularization-based and replay-based methods on temporal sequences from mobile and embedded sensing applications. [21] uses a recurrent neural network (RNN) to evaluate a variety of generic CIL methods on simple TS datasets, such as Stroke-MNIST [24] and AudioSet [22]. The results of these works showcase the effectiveness of using generic CIL methods to mitigate catastrophic forgetting on TS data.

Beyond adapting existing methods from image domain, innovative CIL algorithms for temporal data have also been proposed. [19] and [82] focus on RNN architectures and propose specific regularization-based CIL algorithms. DT^{2}W [54] proposes a novel knowledge distillation (KD) [27] strategy based on soft-DTW [15] to mitigate stability-plasticity dilemma.

A multitude of approaches are structured around experience replay (ER) [10, 59]. CLOPS [35] is a ER-based method for cardiac arrhythmia diagnosis, which includes an importance-based storage strategy and an uncertainty-based retrieval policy for memory buffer management. For efficient audio classification, [37] introduces a fast variant of iCaRL by replacing herding selection with KNN and utilizes quantization to compress memory samples. Using a frozen feature extractor, MAPIC [67] combines a prototype enhancement module with a distance-based classifier for few-shot CIL on medical data.

In the realm of generative replay, [75] continuously trains an autoencoder with Gaussian Mixture Models (GMM) to generate pseudo audio spectrogram data for incremental sound classification. [25] employs separate independent generators for each task, accommodating variable input dimensions in different tasks. [62] trains separate WaveGAN [18] models for different respiratory sound class and conducts a privacy evaluation on synthetic samples. Methods utilizing feature replay or prototypes are also explored. Using a fixed feature extractor, [41] and [42] update the classifier with prototypes for few-shot class-incremental audio classification.

Lastly, architecture-based techniques are also investigated. Inspired by ExpertGate [3], GIM [14] adopts a cascaded model structure which trains a task-specific RNN module for each new task. Along with RNN, a gating autoencoder is trained for each task to select the corresponding module during prediction. Moreover, [66] propose an expandable framework unifying GEM [46] and Net2Net [12] for RNNs.

Despite the efforts made within the field, a lack of a comprehensive evaluation and comparison across various time series datasets is observed. Moreover, TSCIL suffers from inconsistencies in many crucial aspects, including datasets, learning protocols, evaluation schemes and backbones, etc. Some problematic practices related to data normalization and hyperparameter tuning even violate the fundamental principles of CIL. To address these issues, we develop a standard TSCIL framework to systematically and fairly evaluate different CIL methods on TS data.

4 DEVELOPED EVALUATION FRAMEWORK

4.1 Benchmark Datasets

Our TSCIL benchmarks are established with open-sourced real-world time series datasets. Based on these, our toolkit provides a clear way to customize the CIL settings, including the number of classes per task, or the amount of training samples per class. Nevertheless, we follow a common setting in CIL studies to report the results of the balanced training setting in this paper where the amount of training samples for each class is approximately equal. We emphasize the importance of this presumption for 2 reasons. Firstly, it aligns with most standard benchmarks in conventional CIL research in vision domain [70] and facilitates the use of standard evaluation metrics that could be biased if classes were unbalanced. Secondly, the amount of training samples directly affects the difficulty of learning each class. Such influence can affect the performance beyond the CIL algorithm itself, therefore it is beyond the scope of this paper.

Based on such considerations, datasets are selected from two TS-related applications: Human Activity Recognition (HAR) and Gesture Recognition. In general, a group of subjects/volunteers are asked to perform various activities or gestures for a fixed time duration. Such datasets are suitable for CIL, since there are sufficient balanced classes for task split. Some works have utilized HAR datasets for CIL [31, 32, 61], but they adopt the pre-processed vectors as input samples. Instead, we directly use raw time series as inputs, focusing exclusively on the TS modality. In our configuration, TS samples of each dataset exhibit a consistent shape, i.e. the sequence length and number of variables remain the same. Table 1 shows an overview of the employed datasets.

1) UCI-HAR [30] contains temporal sequences of the inertial sensors of smartphones when 6 different daily activities are performed. Data are collected in 50Hz, from 30 volunteers in different ages. Sequences are directly used as inputs, which consist of 9 channels with a temporal span of 128 timesteps.

Table 1: Overview of the benchmark datasets. The last column indicates the number of tasks in the *experiment* stream.

Dataset	Shape ($C \times L$)	Train Size	Test Size	# Classes	# Exp Tasks
UCI-HAR	9×128	7352	2947	6	3
UWave	3×315	896	3582	8	4
DSA	45×125	6840	2280	18	6
GRABMyo	28×128	36120	12040	16	5
WISDM	3×200	18184	6062	18	6

2) UWave [44] includes over 4000 samples collected from 8 subjects while generating 8 simple gesture patterns. We utilize the records from the three axes of the accelerometers so that each input sample is a 3-dimensional time series with 315 timesteps.

3) DSA [4] collects motion sensor segments of 19 daily sports activities carried out by 8 volunteers. Each segment, serving as a sample, is recorded across 45 distinct channels with 125 time steps. To make classes be split equally, we choose to utilize 18 classes from this dataset for experiment.

4) GRABMyo [53] is a large-scaled Surface Electromyography (sEMG) database for hand-gesture recognition. It captures signals during the execution of 16 distinct gestures performed by 43 participants over three separate sessions. All recordings are 5 seconds in duration, collected from 28 channels, and sampled at 2048 Hz. We select one session's data across all subjects for experiment. We first downsample the signal to 256 Hz, followed by the application of a non-overlapping sliding window operation to cut the signal into different samples. Each window of length 0.5 second containing 128 time steps is used as an input sample. We aggregate all of the windows from each subject and perform train-test split with a 3:1 ratio, ensuring that both training and test data are from all the subjects. This avoids introducing distribution shifts caused by subjects between train and test data, appropriate for our focus on CIL. The Offline results in Table 3 indicate that our processed samples contain sufficient information for class differentiation.

5) WISDM [78] is a sensor-based HAR dataset encompassing 18 activities and involving 51 subjects. Following [82], we utilize the phone accelerator modality and extract samples by applying a non-overlapping sliding window with a window size of 200. Each sample comprises a 10-second time series with a frequency of 20 Hz. Similar to the practice for GrabMyo, the dataset is divided into training and test sets with a 3:1 ratio, making both sets include data from all the subjects.

4.2 Learning Protocols

4.2.1 Task Split. Following the standard CIL definition, we need to split the dataset into T tasks, ensuring that each task contains mutually exclusive classes. Similar to the procedure in [57], we shuffle the class order before splitting. That enables us to assess the robustness of CIL methods against class order. After that, we split all the classes equally into each task. Similar to Split-MNIST and Split-CIFAR10 [69], we allocate 2 distinct classes to each task within this work.

4.2.2 Data Normalization. Normalization of input data is crucial for model's training. Many TSCIL studies apply Z-score normalization before task split and use the statistics calculated on the entire dataset for normalization [11, 36, 82]. This practice violates the fundamental principle of CL as the full dataset is not accessible prior to training. To address this issue, we apply instance-wise normalization by inserting an input normalization layer before the first layer of the model. It can be LayerNorm (LN) [6] or InstanceNorm (IN) [68], *without* incorporating a learnable affine transformation or bias. This ensures inputs are normalized to have a mean of 0 and a standard deviation of 1 along specific dimensions. The choice of input normalization layer can be decided based on the performance on the validation tasks. Except for WISDM, where no normalization is applied, we apply IN for UWave, while LN is employed for the remaining datasets.

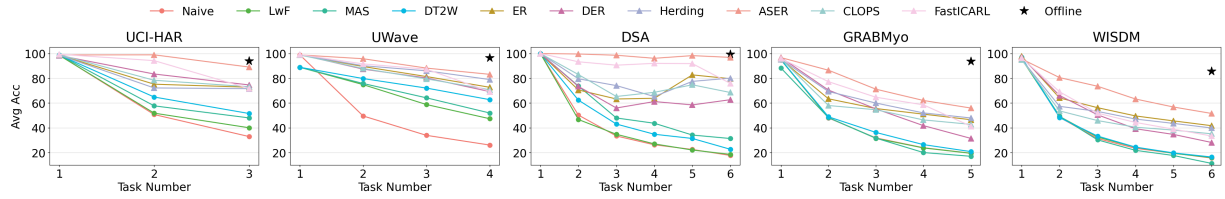
4.2.3 Hyperparameter Tuning. The selection of hyperparameters is a challenging issue in the realm of CL, typically following two protocols. The first [34, 64] involves dividing each task into train, validation, and test sets, then performing grid search. The best parameters are chosen based on validation set performance across all tasks. However, this method, requiring visits to the entire task stream and necessitates a strong assumption of the relatedness between the prior validation data and future tasks. Another protocol [9] divides tasks into a 'validation' stream for cross-validation and hyperparameter tuning, and an 'experiment' stream for training and evaluation. We use the first protocol for UCI-HAR and UWave (having only 3 and 4 tasks, respectively) and the second for datasets with more tasks, setting the validation stream task count to 3. We highlight that both protocols are common standard practices, each with its own advantages and limitations. We offer both options in our toolkit, allowing users to choose based on their needs.

4.3 Selected Methods

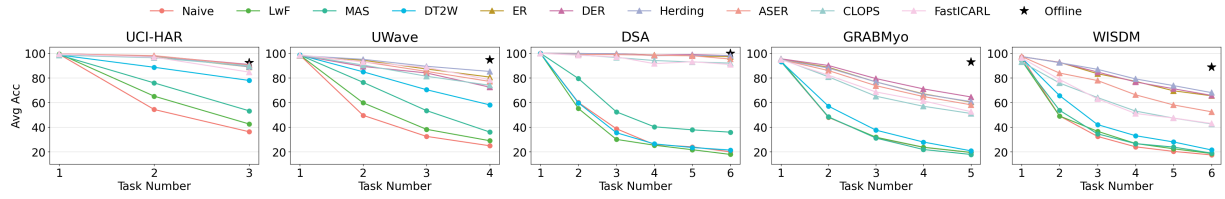
We firstly select 9 representative methods based on regularization and experience replay techniques for comparison. These methods include generic methods proposed in image domain as well as specific algorithms for TS data. Among the regularization-based methods, we choose **LwF** [43], **MAS** [2] and **DT^{2}W** [54]. For experience replay, **ER** [59], **DER** [7], **Herding** [57], **ASER** [63], **CLOPS** [35] and **FastICARL** [37] are included. To investigate the scenario with data privacy concerns, we further incorporate a generative-replay-based method: **GR** [64]. This method avoids saving raw samples, and its experiment results are discussed in Section 5.2. Lastly, we report the results of two straightforward baselines: **Naive** and **Offline**. The former gives the lower bound of performance, as it finetunes the

Table 2: Summary of the implemented CIL algorithms.

Algorithm	Application	Category	Characteristics
LwF [43]	Visual	Regularization	KD on logits
MAS [2]	Visual	Regularization	Parameter regularization
DT^{2}W [54]	Time Series	Regularization	KD on feature maps
ER [59]	Visual	Experience	Replay baseline
Herding [57]	Visual	Experience	Memory update
DER [7]	Visual	Experience	Replay with logits
ASER [63]	Visual	Experience	Memory retrieval
CLOPS [35]	Time Series	Experience	Update & Retrieval
FastICARL [37]	Time Series	Experience	Memory Update
GR [64]	Visual	Generative	Auxiliary generator



(a) BatchNorm



(b) LayerNorm

Figure 2: Evolution of Average Accuracy ($\mathcal{A}_i$) when using (a) BatchNorm or (b) LayerNorm for normalization. Methods utilizing memory buffer are marked with triangles. Since Offline represents joint training on the entire task sequence, its result shows as a single point instead of a curve.

model on tasks sequentially without using any CIL technique. The later one serves as the ideal upper bound since it undergoes joint training with all samples from the entire data stream. A summary of selected CIL methods is presented in Table 2. Additional details are included in Appendix A.1.

4.4 Evaluation Metrics

We employ 3 standard metrics for TSCIL evaluation. Let $a_{i,j}$ represent the accuracy evaluated on the test set of learned task $j \leq i$ upon trained task i. (1) **Average Accuracy** after learning task i is defined as $\mathcal{A}_i = \frac{1}{i}\sum_{j=1}^{i} a_{i,j}$, which is the mean of accuracies across all test sets of learned tasks and reflects the model's overall performance. (2) **Average Forgetting** [9] after leaning task i is defined as $\mathcal{F}_i = \frac{1}{i-1}\sum_{j=1}^{i-1} f_{i,j}$, where $f_{i,j} = \max_{k\in\{1,\ldots,i-1\}}(a_{k,j}) - a_{i,j}, j < i$ represents how much performance degrades on task j due to learning task i. This metric reflects how much of the acquired knowledge the model has forgotten in a task-level. (3) **Average Learning Accuracy** [58] is defined as $\mathcal{A}_{cur} = \frac{1}{T}\sum_{i=1}^{T} a_{i,i}$. This metric indicates the overall impact of using a CIL method on learning new tasks, which is reflected by the average of *current task accuracy* $a_{i,i}$ across all the tasks in sequence. To reflect the final performance, the community commonly reports **Final Average Accuracy** $\mathcal{A}_T$ and **Final Average Forgetting** $\mathcal{F}_T$, which are computed across all the tasks after the final task has been learned.

4.5 Model Architecture

For the experiments in this paper, we employ a **1D-CNN** backbone similar to [56] as the feature extractor. It consists of four convolutional blocks, with each block comprising a 1D-Convolutional layer, a BatchNorm (BN) layer, a MaxPooling layer and a Dropout Layer. Unless otherwise stated, we utilize a single-head classifier with softmax activation across all the algorithms. We specifically investigate the impact of different types of classifier in the ablation study. For methods using memory buffer, we set the buffer size to 5% of the training size in the experiment task stream. Furthermore, normalization layers also play a vital role in CIL problems. Although most literature incorporate BN layers into their models, it has been empirically shown that BN layers suffer from the biased issue in CIL scenarios [52]. We further investigate this issue in the realm of TSCIL by comparing the results of using BN and LN. For the generator of GR, we utilize a TimeVAE [17], with both the encoder and decoder designed with four layers of Conv1D and ConvTranspose1d, respectively.

4.6 Implementation Details

All the experiments are run 5 times with different class orders and random seeds. For each run, we tune its particular best hyperparameters as the protocols described above. Similar to [70], all models are trained for 100 epochs per task using an Adam optimizer with a learning rate of 0.001 and a batch size of 64. The learning rate scheduler is configured as a hyperparameter for tuning. To alleviate overfitting on training data, early stopping is used during training. For a fair comparison, we choose not to tune the architecture-related parameters for different methods. Instead, we employ a fixed and consistent model architecture. Further details on implementation of the framework can be found in Appendix A.2. We highlight that our framework is extensible. Users can follow the instructions in our code page to incorporate new datasets, algorithms, and custom experimental setups.

5 EXPERIMENTS AND DISCUSSION

5.1 Evaluation of Regularization-based and ER-based Methods

5.1.1 Performance comparison using BN and LN. We first focus on a basic scenario in which saving historical samples is permitted. As listed in Table 2, we evaluate 3 regularization-based methods and 5 methods rooted on experience replay. Simultaneously, we also investigate the influence of normalization layers by running 2

Table 3: Evaluation metrics of regularization-based and ER-based methods on our 5 TSCIL benchmarks when using (a) BatchNorm or (b) LayerNorm for normalization. Metrics introduced in Section 4.4 are reported, which are $\mathcal{A}_T(\uparrow)$, $\mathcal{F}_T(\downarrow)$ and $\mathcal{A}_{cur}(\uparrow)$. For each metric, its mean and confidence interval on 5 runs are reported.

(a) BatchNorm

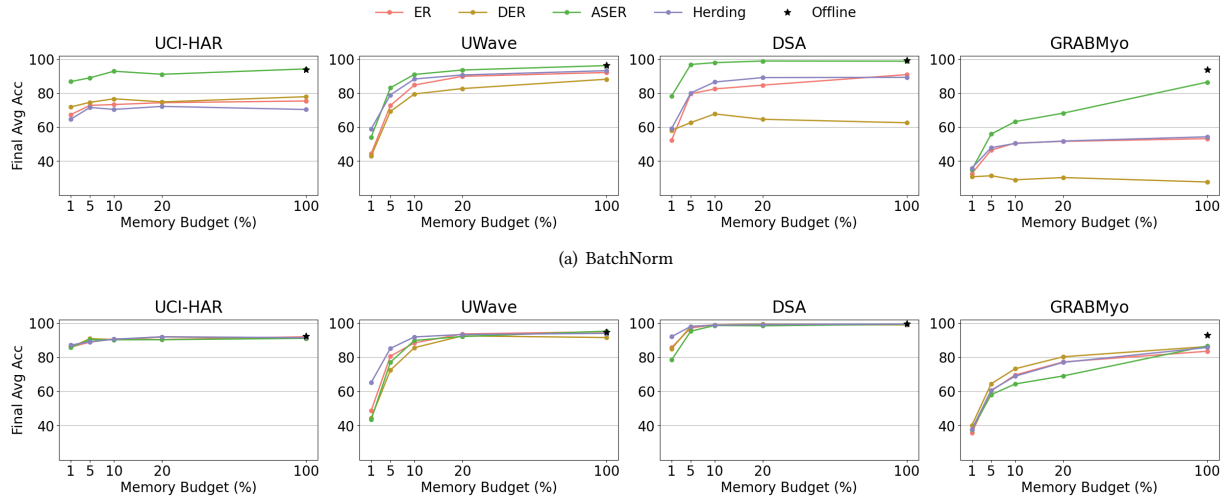
Dataset	Metric	Naive	Offline	LwF	MAS	DT^{2}W	ER	DER	Herding	ASER	CLOPS	FastICARL
UCI-HAR	$\mathcal{A}_T$	32.9±1.1	93.9±0.6	40.0±7.4	48.0±11.4	**51.5**±14.1	72.8±19.7	74.6±3.5	71.6±23.9	**89.1**±6.1	73.2±10.5	71.7±19.1
	$\mathcal{F}_T$	97.8±3.3	N.A	83.0±14.1	71.6±11.8	**62.0**±16.3	22.1±37.2	16.6±18.8	25.1±44.8	**13.2**±11.4	18.4±25.6	32.4±34.0
	$\mathcal{A}_{cur}$	98.1±3.3	N.A	**95.4**±5.6	93.2±9.5	92.8±6.2	87.2±10.4	85.6±11.7	88.3±9.0	**97.8**±1.9	85.2±7.7	93.3±5.7
UWave	$\mathcal{A}_T$	26.0±3.0	96.6±0.7	47.3±11.1	51.9±11.1	**62.7**±11.6	72.7±2.7	69.2±4.5	79.0±1.1	**83.2**±3.5	70.8±3.0	69.3±2.4
	$\mathcal{F}_T$	97.3±3.7	N.A	51.7±19.2	45.7±15.1	**32.9**±12.0	33.6±3.4	37.8±6.9	24.7±2.4	**20.3**±4.5	36.1±3.7	37.8±4.2
	$\mathcal{A}_{cur}$	99.0±0.6	N.A	**86.1**±14.1	78.3±16.2	83.6±17.0	97.9±0.5	97.6±1.0	97.5±1.4	**98.4**±0.4	97.8±0.4	97.6±0.9
DSA	$\mathcal{A}_T$	17.6±2.7	99.5±0.6	18.5±6.4	**31.3**±6.4	22.6±6.6	79.6±15.6	62.7±23.0	80.1±6.0	**96.9**±2.2	68.5±20.5	75.9±21.1
	$\mathcal{F}_T$	98.8±3.2	N.A	85.1±22.7	**60.0**±11.5	87.5±8.2	23.6±19.2	21.4±17.4	20.4±6.5	**3.7**±2.7	32.1±27.4	25.1±26.4
	$\mathcal{A}_{cur}$	100.0±0.0	N.A	87.8±29.0	81.2±16.3	**95.5**±1.7	89.9±9.6	76.9±20.0	90.9±7.1	**100.0**±0.1	88.0±9.0	95.3±1.2
GRABMyo	$\mathcal{A}_T$	19.4±0.3	93.8±1.0	19.4±0.6	16.9±2.2	**20.7**±5.2	46.5±3.2	31.4±3.7	47.9±3.9	**55.9**±5.1	42.7±4.3	41.3±5.3
	$\mathcal{F}_T$	95.4±1.6	N.A	95.0±1.5	**49.8**±25.3	59.3±19.0	35.0±5.7	59.8±8.2	33.1±8.7	49.4±5.0	**31.1**±4.9	47.0±5.8
	$\mathcal{A}_{cur}$	95.8±1.2	N.A	**95.4**±1.1	56.8±20.9	67.2±17.7	73.7±4.7	79.2±7.3	73.2±6.2	**95.4**±1.3	65.1±5.6	78.4±5.9
WISDM	$\mathcal{A}_T$	15.5±1.2	85.7±1.9	15.9±0.7	11.2±5.3	**16.4**±6.6	41.7±5.0	28.1±10.1	39.9±10.6	**51.6**±13.7	35.1±6.1	33.4±4.8
	$\mathcal{F}_T$	95.6±3.4	N.A	96.4±2.7	78.0±8.2	**63.8**±44.0	**27.9**±12.4	55.6±24.1	28.6±19.7	53.1±14.3	33.7±13.7	39.7±6.8
	$\mathcal{A}_{cur}$	95.2±3.1	N.A	**96.2**±1.8	76.2±5.4	69.3±38.2	60.7±10.2	74.5±18.0	61.3±10.1	**95.8**±2.0	60.3±10.3	66.2±5.8

(b) LayerNorm

Dataset	Metric	Naive	Offline	LwF	MAS	DT^{2}W	ER	DER	Herding	ASER	CLOPS	FastICARL
UCI-HAR	$\mathcal{A}_T$	36.2±10.9	92.5±0.8	42.5±14.2	53.2±7.5	**77.9**±15	88.9±2.7	**90.9**±1.7	89.0±1.9	90.3±1.9	89.8±1.5	84.7±4.8
	$\mathcal{F}_T$	92.5±13.5	N.A	83.1±19.0	64.8±18.8	**9.3**±6.6	10.8±6.8	**8.5**±5.7	10.7±6.5	9.6±5.7	9.2±4.9	18.2±10.7
	$\mathcal{A}_{cur}$	97.9±3.5	N.A	**97.9**±3.8	95.8±4.5	83.7±19.7	96.1±2.8	96.3±2.7	96.1±3.2	**96.6**±1.8	95.9±2.6	96.8±3.6
UWave	$\mathcal{A}_T$	24.8±0.1	94.7±0.7	28.9±4.6	36.0±8.1	**58.1**±18.3	80.7±1.4	72.4±18.0	**85.2**±2.2	77.1±7.5	74.0±2.1	78.5±0.8
	$\mathcal{F}_T$	98.4±1.5	N.A	69.8±38.6	71.5±11.1	**35.4**±25.6	22.7±1.3	33.5±22.1	**16.6**±2.4	28.1±9.1	32.2±3.2	26.4±0.8
	$\mathcal{A}_{cur}$	98.6±1.1	N.A	81.2±25.0	**89.7**±4.2	87.3±2.6	97.7±1.0	97.5±1.5	97.5±1.4	**98.2**±0.9	98.1±0.7	98.3±0.4
DSA	$\mathcal{A}_T$	19.9±5.1	99.8±0.1	17.8±4.2	**35.9**±5.6	21.4±7.1	97.2±2.2	**98.0**±1.0	98.0±1.8	95.3±4.7	92.0±4.1	90.8±3.0
	$\mathcal{F}_T$	96.1±6.2	N.A	89.9±12.7	**53.0**±9.1	94.0±8.7	3.3±2.7	**2.2**±1.2	2.3±2.2	5.6±5.7	9.4±5.0	10.9±3.6
	$\mathcal{A}_{cur}$	100.0±0.0	N.A	92.5±12.3	80.0±9.5	**99.7**±0.6	99.9±0.1	99.9±0.2	99.9±0.1	99.9±0.2	99.9±0.2	99.9±0.1
GRABMyo	$\mathcal{A}_T$	19.4±0.3	92.9±2.3	19.5±0.2	17.8±1.2	**20.8**±7.1	60.3±1.1	**64.5**±3.5	60.7±2.7	58.1±3.9	51.1±3.5	52.6±4.2
	$\mathcal{F}_T$	94.4±2.4	N.A	94.7±2.6	84.2±3.5	**21.9**±10.4	42.1±1.6	**36.6**±3.4	41.7±3.3	45.3±4.7	53.6±4.5	52.1±5.6
	$\mathcal{A}_{cur}$	94.9±1.7	N.A	**95.2**±1.9	85.2±3.0	36.0±11.2	94.0±0.8	93.8±0.8	94.1±1.4	**94.4**±1.1	93.9±1.4	94.2±2.0
WISDM	$\mathcal{A}_T$	17.4±3.4	88.7±1.8	18.5±4.3	18.9±5.0	**21.4**±6.6	65.4±5.8	65.8±6.9	**68.0**±7.6	52.4±17.1	42.6±7.7	42.9±5.1
	$\mathcal{F}_T$	95.4±5.5	N.A	91.9±7.5	75.6±12.2	**41.2**±21.5	34.7±5.4	35.2±7.3	**31.9**±7.9	52.4±20.6	62.3±8.2	57.3±9.2
	$\mathcal{A}_{cur}$	97.0±1.6	N.A	**95.1**±3.8	81.8±7.7	49.4±22.1	94.3±1.9	95.1±1.2	94.6±1.2	**96.1**±1.2	94.5±1.2	90.7±6.3

sets of experiments. One uses the default CNN backbone with BN layers, the other replaces the BN layers in the model with LN layers. The results of overall performances are shown in Table 3(a) for BN and Table 3(b) for LN. We also present the evolution of Average Accuracy $\mathcal{A}_i$ across tasks in Figure 2. The evaluation results answer the following questions.

Question 1: How do regularization vs ER perform in TSCIL? Similar to the findings in image domain [39, 48], *ER-based methods stably outperform regularization-based methods without saving exemplars in TSCIL*. As anticipated, without using any CIL techniques, Naive inevitably results in catastrophic forgetting. By saving memory samples, all the ER-based methods effectively mitigate forgetting across datasets. Surprisingly, when using LN for normalization, even the basic ER method with a 5% memory budget can sometimes achieve close results to the offline training upper bound. In contrast, regularization-based methods only show clear benefits in simpler benchmarks with fewer tasks like UCI-HAR and UWave. In these datasets, DT^{2}W consistently outperforms MAS, which in turn offers better results than LwF. However, in more challenging benchmarks, the same regularization approaches fail



(a) BatchNorm

(b) LayerNorm

Figure 3: Evolution of Average Accuracy ($\mathcal{A}_i$) when using different memory budget. The results encompass 4 ER-based methods on 4 datasets, utilizing BatchNrom (top row) or LayerNorm (bottom row) for normalization.

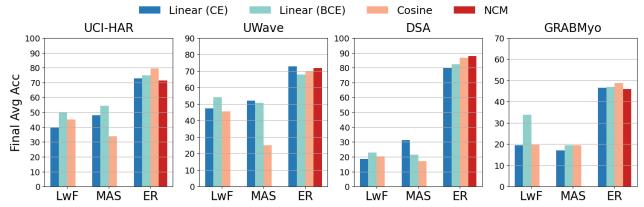
almost completely. Specifically, they struggle with balancing stability and plasticity, leading to either significant forgetting (LwF) or compromised learning accuracy (MAS and DT^{2}W).

Question 2: How does the choice of BN and LN affects TSCIL? While the choice between BN and LN has a marginal impact on offline training, we find that *using LN appears to significantly improve the performance for most ER-based methods*. Remarkably, the impact of switching to LN is so profound that it can overshadow the choice of the algorithm itself. In some cases, merely transitioning from BN to LN within the same algorithm can boost the performance to levels almost on par with offline training. [52] attributes this phenomenon to the bias of running statistics in BN, which is cased by the imbalance of new and memory samples and results in a loss of previously acquired knowledge. In contrast, employing instance-wise normalization, such as LN, effectively circumvents this issue. However, we emphasize that the influence of the bias of BN is *bidirectional*. Based on the change of learning accuracy $\mathcal{A}_{cur}$, we find that the bias of BN not only degrades the stability, but also hinders the learning of new knowledge. Additionally, the bias of BN exerts a more pronounced effect on replay with logits compared to replay with raw samples, with DER experiencing significant improvement upon substituting BN with LN. Interestingly, ASER appears as an exemption: its performance on BN is extensively better than other compared methods, yet it does not show notable benefit from using LN. We posit that this is due to the *MemoryRetrieval* mechanism of ASER, which selects a balanced and representative batch of memory samples to maintain an unbiased statistics in BN layers. In some extent, the superiority of ASER on BN underscores the significance of *MemoryRetrieval* within ER techniques. Contrary to ER-based methods, regularization-based methods fail to exhibit a consistent pattern across BN and LN. In conclusion, ER-based methods consistently benefit from using LN with a substantial improvement, whereas regularization-based methods need to decide the choice of BN or LN based on the dataset.

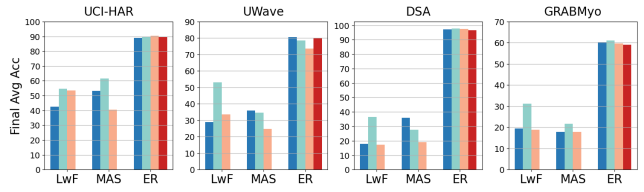
5.1.2 Ablation study. In this section, we investigate the effect of memory budget and classifier type on TSCIL performance. We first evaluate ER-based methods across a range of memory budgets, with results shown in Figure 3. The memory budgets are set to 1%, 5%, 10%, 20% and 100% of the size of whole training dataset. After that, we compare the performance of LwF, MAS, ER when using 3 different types of classifier. We present the results in Figure 4. All the evaluations are conducted using both BN and LN for normalization. The results answer the following questions.

Question 3: How does memory budget affect ER-based methods? As intuitively expected, ER-based methods generally demonstrate increased performance as the memory buffer size grows. However, it's important to note that *the performance gains saturate beyond a certain buffer size*. Such trends exhibit differences between using BN and LN. A surprising observation is at a 100% memory budget where all encountered data is saved for replay (same as Offline). When using BN, all methods except for ASER show an obvious performance gap compared to offline training. This suggests that the bias of BN towards new tasks does not stem from the imbalance between the memory budget and the size of the new task's training data. Rather, *the bias arises from the 2-batch ER pipeline* (see Algorithm 1 in Appendix A.1). As the number of tasks increases, the proportion of each old class samples in $B_{\mathcal{M}}$ diminishes, leading to an unbalanced distribution of old and new class samples in each step. In contrast, the results on LN show consistent trends in all the configurations, saturating at a level that closely approximates Offline. These outcomes demonstrate a potential issue in the standard ER protocol and further underscore the advantages of using LN for replay in TSCIL.

Question 4: How do different classifier types affect TSCIL? The conventional softmax classifier is known to exhibit a bias problem in CIL scenarios without rehearsal data, that is, the magnitudes of weights of new classes are larger than those of old classes [28].



(a) BatchNorm



(b) LayerNorm

Figure 4: Ablation study on different types of classifiers. The first two represent the single-head classifier trained with CE and BCE, respectively.

The reason is that minimizing softmax classification loss always reduces the weight magnitudes of the old classes. A simple approach to handle this issue is to replace softmax with sigmoid and train the model with BCE [57, 65]. Using such a BCE-based classifier, we observe that the results of LwF are consistently improved with a notable margin. However, this improvement is not consistently observed in MAS or ER. Another classifier is Split Cosine Classifier [28], which normalizes the features and class weights and compute their cosine similarity. However, employing such classifier doesn't improve the performance consistently and may even hinder MAS. Lastly, the NCM classifier is suited only for methods utilizing memory samples, and no significant improvement is observed. We posit the reason is that the rehearsal of memory samples mitigates the bias in the single-headed classifier,resulting in a similar performance across different classifiers. In summary, the choice of classifier depends on methods and datasets, and it is less critical for methods employing ER.

5.2 Evaluation of GR in Privacy-Sensitive Scenarios

In this section, we consider a practical scenario with data privacy concerns, restricting the storage of raw historical samples. As exemplar-free regularization-based approaches exhibit inherent limitations, we investigate GR by using a TimeVAE [17] as the generator. Experiments are run on models utilizing BN or LN, with results presented in Table 4.

Question 5: How does GR vs ER perform in TSCIL? In simpler datasets like UCI-HAR and UWave, GR demonstrates substantial efficacy as an alternative to ER. Notably, it consistently outperforms regularization-based approaches and also exhibits comparable or better results to ER. By displaying raw and generated samples from UWave, we find that GR can produce pseudo samples that resemble certain patterns found in the original data. (see Figure 7). However, the effectiveness of GR is limited in more complex datasets like DSA and GRABMyo, exhibiting a notable performance gap compared to ER. We attribute GR's limitations to two reasons. First, training a proficient generator model on datasets with large number of classes or variables is still challenging for time series data, especially when the training process is incremental. The diversity of generated samples is also limited (see Figure 6 in Appendix B.1). Secondly, the naive GR method cannot control the class of generated samples, hindering a balanced rehearsal on old classes. In contrast, ER's i.i.d. memory updating circumvents these issues, leading to a pronounced performance differential. Additionally, similar to ER, GR benefits from the use of LN over BN, especially in UCI-HAR and DSA. This indicates that inherent bias impacts all CIL methods employing replay. In summary, while GR shows strong competitiveness in simpler datasets, it encounters notable challenges in more complex environments.

Table 4: Evaluation metrics of GR on 4 TSCIL benchmarks.

Norm	Metric	Dataset			
		UCI-HAR	UWave	DSA	GRABMyo
BatchNorm	$\mathcal{A}_T$	$62.2_{\pm 11.1}$	$75.5_{\pm 8.2}$	$37.2_{\pm 5.9}$	$22.4_{\pm 3.9}$
	$\mathcal{F}_T$	$45.9_{\pm 19.6}$	$27.8_{\pm 10.9}$	$67.3_{\pm 11.8}$	$69.1_{\pm 6.6}$
	$\mathcal{A}_{cur}$	$92.8_{\pm 10.4}$	$96.3_{\pm 1.6}$	$93.2_{\pm 6.0}$	$77.6_{\pm 4.1}$
LayerNorm	$\mathcal{A}_T$	$81.0_{\pm 6.7}$	$75.8_{\pm 10.3}$	$66.7_{\pm 14.3}$	$20.3_{\pm 0.7}$
	$\mathcal{F}_T$	$24.7_{\pm 14.7}$	$29.3_{\pm 12.9}$	$39.9_{\pm 17.2}$	$94.9_{\pm 2.8}$
	$\mathcal{A}_{cur}$	$97.5_{\pm 3.8}$	$97.8_{\pm 0.8}$	$99.9_{\pm 0.1}$	$96.2_{\pm 1.6}$

5.3 Analyzing Intra-Class Variations among Subjects

Question 6: How do intra-class variations affect TSCIL? Time series data are commonly collected from various subjects or sources, each may exhibit a distinct input domain. For instance, Figure 5 depicts the feature distribution within a VAE trained on two classes from DSA. Notably, each class forms eight clusters within the feature space, each corresponding to a different subject. Although such phenomenon is often overlooked in TSCIL, we find that the distribution shift between different subjects may impact the learning performance in a non-trivial level. We further analyse how such subject distribution affects ER-based methods in Appendix B.2.

To further investigate this, we use the DSA dataset to compare the original ER baseline with two of its variants. The original ER employs reservoir sampling for *MemoryUpdate*, theoretically ensuring that memory samples in the buffer are i.i.d to the original distribution. However, its *MemoryRetrieval* policy, based on random selection, may not ensure that each batch of replayed samples follows the subject distribution. Our first variant modifies the *MemoryUpdate* policy to select samples only from part of the subjects, deliberately causing the subject distribution of memory samples to deviate from the actual distribution. The second variant maintains the original *MemoryUpdate* policy but improves the *MemoryRetrieval* policy to ensure memory samples are subject-balanced in each retrieved batch. The evaluation metrics are presented in Table 5, where the first two methods correspond to the first variant sampling from two and four subjects, respectively. "Balanced" represents the second balanced-retrieval variant.

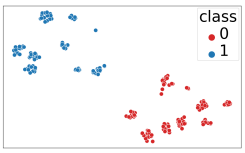


Figure 5: Intra-class variations in DSA.

Method	$\mathcal{A}_T$	$\mathcal{F}_T$	$\mathcal{A}_{cur}$
ER (2 Sub)	73.1$_{\pm 25.7}$	24.1$_{\pm 26.1}$	85.8$_{\pm 14.4}$
ER (4 Sub)	76.4$_{\pm 14.4}$	25.8$_{\pm 14.1}$	92.4$_{\pm 6.3}$
ER	79.6$_{\pm 15.6}$	23.6$_{\pm 19.2}$	89.9$_{\pm 9.6}$
Balanced	**87.8**$_{\pm 12.1}$	**13.3**$_{\pm 14.0}$	**93.9**$_{\pm 7.1}$

Table 5: Evaluation of subject-based ER variants using BN.

The observed results underscore the significance of maintaining the subject distribution in TSCIL. Specifically, sampling from part of the subjects demonstrates a diminished replay effect. In contrast, the use of subject-balanced memory samples significantly enhances the rehearsal process. This finding corroborates the idea that incorporating intra-class variations into CIL can improve outcomes. Ignoring this aspect, on the other hand, leads to sub-optimal results. These insights point to an emerging challenge in TSCIL, particularly for methods relying on ER and GR: the need to account for distribution shifts within class caused by different input domains.

6 FUTURE DIRECTIONS

This section outlines potential future directions in TSCIL research.

(1) *Generative Replay for complex time series*: Using GR in complex datasets is a challenge for further exploration. We list several potential solutions. The first is to convert raw time series into time-frequency representations like spectrograms, and to use image-generating models to improve TS synthesis [1, 26, 76]. The second is to apply causality learning, which aims to uncover the underlying data generation processes. Combining it with continual learning emerges as a promising approach to enhance model interpretability and adaptability to distribution shifts [13], especially when applying it to time series generation [80]. The last one is to investigate CIL methods based on *model inversion* [45, 81], which have proven effective in synthesizing pseudo samples in the image domain .

(2) *Intra-class variations*: The incorporation of intra-class variations into CIL methods are different based on the strategies they use. For regularization, assuming that intra-class variations are similar across different classes [83], a potential avenue is to design a metric to capture the intra-class variations that can be used as a regularization term. For ER, a direction is to tailor memory management policies to take intra-class variations into account. For GR, one may implement an intra-cluster-conditional generator [84] to improve the performance.

(3) *Non-standard CIL setup*: This paper focuses on the standard CIL setup. For industry settings, one may need to consider more practical factors beyond our current academic setup, such as data imbalances [35], irregular sampling [25], and online [48] or mutliview learning [38]. We plan to extend our framework to incorporate these challenging settings in the future.

(4) *Incorporation of frequency domain knowledge*: One of critical limitations of current CIL methods lies in the overlook of the intrinsic differences between TS and images. For example, time series are more likely to exhibit periodicity than images. Moreover, TS encapsulates crucial information within the frequency domain or the time-frequency domain. However, existing methods are generic, overlooking these vital properties. Incorporating such properties into TS-specific algorithms is an important topic for future research.

(5) *Time Series Foundation Model*: Large pretrained models has shown competitive performance in image-based CIL [77], even in the absence of memory samples. However, the exploration of pretrained models in TSCIL remains understudied, primarily due to the absence of a universal TS pretrained model. However, the recent advancements in developing Time Series Foundation Models [79, 86] mark a significant milestone. Such models are pretrained on a vast collection of TS datasets, which can be applied to various downstream tasks like classification or forecasting. Applying such models for TSCIL is a promising direction to explore.

7 CONCLUSION

This paper introduces a unified evaluation framework for Time Series Class-incremental Learning (TSCIL). We provide a holistic comparison to demonstrate the promises and limitations of existing CIL strategies in addressing TSCIL problem. Our extensive experiments evaluate important aspects of TSCIL, including algorithms, normalization layers, memory budget, and the classifier choice. We find that replay-based methods generally demonstrate superiority than regularization techniques, and using LayerNorm instead of BatchNorm significantly alleviates the stability-plasticity dilemma. We further explore some challenges of time series data that are vital to the success of TSCIL. Results and analysis highlight the challenges of normalization, data privacy and intra-class variation, and how they impact the results of TSCIL. We firmly believe our work provides a valuable asset for the TSCIL research and development communities.

ACKNOWLEDGMENTS

This research is part of the programme DesCartes and is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

REFERENCES

[1] Ahmed Alaa, Alex James Chan, and Mihaela van der Schaar. 2021. Generative Time-series Modeling with Fourier Flows. In *International Conference on Learning Representations*.
[2] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *ECCV*. 139–154.
[3] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In *CVPR*. 3366–3375.
[4] Kerem Altun and Billur Barshan. 2010. Human activity recognition using inertial/magnetic sensor units. In *International workshop on human behavior understanding*. Springer, 38–51.
[5] Jacob Armstrong and David A Clifton. 2022. Continual learning of longitudinal health records. In *2022 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*. IEEE, 01–06.
[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
[7] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. 2020. Dark experience for general continual learning: a strong, simple baseline. *NIPS* 33 (2020), 15920–15930.
[8] Antonio Carta, Lorenzo Pellegrini, Andrea Cossu, Hamed Hemati, and Vincenzo Lomonaco. 2023. Avalanche: A PyTorch Library for Deep Continual Learning. *Journal of Machine Learning Research* 24, 363 (2023), 1–6. http://jmlr.org/papers/v24/23-0130.html
[9] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018. Efficient Lifelong Learning with A-GEM. In *ICLR*.
[10] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, P Dokania, P Torr, and M Ranzato. 2019. Continual learning with tiny episodic memories. In *Workshop on Multi-Task and Lifelong Reinforcement Learning*.

[11] Jagmohan Chauhan, Young D Kwon, Pan Hui, and Cecilia Mascolo. 2020. Contauth: Continual learning framework for behavioral-based user authentication. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 4 (2020), 1–23.

[12] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. 2016. Net2Net: Accelerating Learning via Knowledge Transfer. arXiv:1511.05641 [cs.LG]

[13] Nikhil Churamani, Jiaee Cheong, Sinan Kalkan, and Hatice Gunes. 2023. Towards Causal Replay for Knowledge Rehearsal in Continual Learning. In *Proceedings of The First AAAI Bridge Program on Continual Causality (Proceedings of Machine Learning Research, Vol. 208)*, Martin Mundt, Keiland W. Cooper, Devendra Singh Dhami, Adéle Ribeiro, James Seale Smith, Alexis Bellot, and Tyler Hayes (Eds.). PMLR, 63–70.

[14] Andrea Cossu, Antonio Carta, and Davide Bacciu. 2020. Continual learning with gated incremental memories for sequential data processing. In *IJCNN*. IEEE, 1–8.

[15] Marco Cuturi and Mathieu Blondel. 2017. Soft-dtw: a differentiable loss function for time-series. In *ICML*. PMLR, 894–903.

[16] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *TPAMI* 44, 7 (2021), 3366–3385.

[17] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. 2021. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095* (2021).

[18] Chris Donahue, Julian McAuley, and Miller Puckette. 2019. Adversarial Audio Synthesis. In *ICLR*.

[19] Lea Duncker, Laura Driscoll, Krishna V Shenoy, Maneesh Sahani, and David Sussillo. 2020. Organizing recurrent network dynamics by task-computation to enable continual learning. *NIPS* 33 (2020), 14387–14397.

[20] Ashutosh Dhar Dwivedi, Gautam Srivastava, Shalini Dhar, and Rajani Singh. 2019. A decentralized privacy-preserving healthcare blockchain for IoT. *Sensors* 19, 2 (2019), 326.

[21] Benjamin Ehret, Christian Henning, Maria Cervera, Alexander Meulemans, Johannes Von Oswald, and Benjamin F Grewe. 2021. Continual learning in recurrent neural networks. In *ICLR*.

[22] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *ICASSP*. IEEE, 776–780.

[23] Stephen T Grossberg. 2012. *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*. Vol. 70. Springer Science & Business Media.

[24] Caglar Gulcehre, Sarath Chandar, and Yoshua Bengio. 2017. Memory augmented neural networks with wormhole connections. *arXiv preprint arXiv:1701.08718* (2017).

[25] Vibhor Gupta, Jyoti Narwariya, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2021. Continual learning for multivariate time series tasks with variable input dimensions. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 161–170.

[26] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. 2017. CNN architectures for large-scale audio classification. In *ICASSP 2017*. 131–135.

[27] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[28] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *CVPR*. 831–839.

[29] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery* 33, 4 (2019), 917–963.

[30] Dewi Pramudi Ismi, Shireen Panchoo, and Murinto Murinto. 2016. K-means clustering based filter feature selection on high dimensional data. *International Journal of Advances in Intelligent Informatics* 2 (2016), 38–45.

[31] Saurav Jha, Martin Schiemer, and Juan Ye. 2020. Continual learning in human activity recognition: An empirical analysis of regularization. *arXiv preprint arXiv:2007.03032* (2020).

[32] Saurav Jha, Martin Schiemer, Franco Zambonelli, and Juan Ye. 2021. Continual learning in sensor-based human activity recognition: An empirical benchmark analysis. *Information Sciences* 575 (2021), 1–21.

[33] Zixuan Ke and Bing Liu. 2022. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701* (2022).

[34] James Kirkpatrick, Razvan Pascanu, et al. 2017. Overcoming catastrophic forgetting in neural networks. *PNAS* 114, 13 (2017), 3521–3526.

[35] Dani Kiyasseh, Tingting Zhu, and David Clifton. 2021. A clinical deep learning framework for continually learning from cardiac signals across diseases, time, modalities, and institutions. *Nature Communications* 12, 1 (2021), 4221.

[36] Young D Kwon, Jagmohan Chauhan, Abhishek Kumar, Pan Hui HKUST, and Cecilia Mascolo. 2021. Exploring system performance of continual learning for mobile and embedded sensing applications. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 319–332.

[37] Young D Kwon, Jagmohan Chauhan, and Cecilia Mascolo. 2021. Fasticarl: Fast incremental classifier and representation learning with efficient budget allocation in audio sensing applications. *arXiv preprint arXiv:2106.07268* (2021).

[38] Depeng Li, Tianqi Wang, Junwei Chen, Kenji Kawaguchi, Cheng Lian, and Zhigang Zeng. 2024. Multi-view class incremental learning. *Information Fusion* 102 (2024), 102021.

[39] Depeng Li, Tianqi Wang, Junwei Chen, Qining Ren, Kenji Kawaguchi, and Zhigang Zeng. 2024. Towards Continual Learning Desiderata via HSIC-Bottleneck Orthogonalization and Equiangular Embedding. *arXiv preprint arXiv:2401.09067* (2024).

[40] Depeng Li and Zhigang Zeng. 2023. CRNet: A Fast Continual Learning Framework With Random Theory. *TPAMI* (2023).

[41] Yanxiong Li, Wenchang Cao, Jialong Li, Wei Xie, and Qianhua He. 2023. Few-shot Class-incremental Audio Classification Using Stochastic Classifier. *arXiv preprint arXiv:2306.02053* (2023).

[42] Yanxiong Li, Wenchang Cao, Wei Xie, Jialong Li, and Emmanouil Benetos. 2023. Few-shot Class-incremental Audio Classification Using Dynamically Expanded Classifier with Self-attention Modified Prototypes. *IEEE Transactions on Multimedia* (2023).

[43] Zhizhong Li and Derek Hoiem. 2016. Learning Without Forgetting. In *ECCV*. Springer, 614–629.

[44] Jiayang Liu, Zhen Wang, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. In *2009 IEEE International Conference on Pervasive Computing and Communications*. 1–9. https://doi.org/10.1109/PERCOM.2009.4912759

[45] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. 2020. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*. 12245–12254.

[46] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *NeurIPS*. 6467–6476.

[47] Reem A Mahmoud and Hazem Hajj. 2022. Multi-objective Learning to Overcome Catastrophic Forgetting in Time-series Applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16, 6 (2022), 1–20.

[48] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. 2022. Online continual learning in image classification: An empirical survey. *Neurocomputing* 469 (2022), 28–51.

[49] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost Van De Weijer. 2022. Class-incremental learning: survey and performance evaluation on image classification. *TPAMI* 45, 5 (2022), 5513–5533.

[50] Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. Vol. 24. Elsevier, 109–165.

[51] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730* (2022).

[52] Quang Pham, Chenghao Liu, and HOI Steven. 2022. Continual Normalization: Rethinking Batch Normalization for Online Continual Learning. In *ICLR*.

[53] Ashirbad Pradhan, Jiayuan He, and Ning Jiang. 2022. Multi-day dataset of forearm and wrist electromyogram for hand gesture recognition and biometrics. *Scientific Data* 9, 1 (2022), 733.

[54] Zhongzheng Qiao, Minghui Hu, Xudong Jiang, Ponnuthurai Nagaratnam Suganthan, and Ramasamy Savitha. 2023. Class-Incremental Learning on Multivariate Time Series Via Shape-Aligned Temporal Distillation. In *ICASSP*. IEEE, 1–5.

[55] Xiangfei Qiu, Jilin Hu, Lekui Zhou, Xingjian Wu, Junyang Du, Buang Zhang, Chenjuan Guo, Aoying Zhou, Christian S Jensen, Zhenli Sheng, et al. 2024. TFB: Towards Comprehensive and Fair Benchmarking of Time Series Forecasting Methods. *arXiv preprint arXiv:2403.20150* (2024).

[56] Mohamed Ragab, Emadeldeen Eldele, Wee Ling Tan, Chuan-Sheng Foo, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, and Xiaoli Li. 2023. Adatime: A benchmarking suite for domain adaptation on time series data. *ACM Transactions on Knowledge Discovery from Data* 17, 8 (2023), 1–18.

[57] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *CVPR*. 2001–2010.

[58] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2018. Learning to Learn without Forgetting by Maximizing Transfer and Minimizing Interference. In *ICLR*.

[59] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. 2019. Experience replay for continual learning. In *NeurIPS*. 350–360.

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

[61] Martin Schiemer, Lei Fang, Simon Dobson, and Juan Ye. 2023. Online continual learning for human activity recognition. *Pervasive and Mobile Computing* (2023), 101817.

[62] Anja Shevchyk, Rui Hu, Kevin Thandiackal, Michael Heizmann, and Thomas Brunschwiler. 2022. Privacy preserving synthetic respiratory sounds for class

incremental learning. *Smart Health* 23 (2022), 100232.
[63] Dongsub Shim, Zheda Mai, Jihwan Jeong, Scott Sanner, Hyunwoo Kim, and Jongseong Jang. 2021. Online Class-Incremental Continual Learning with Adversarial Shapley Value. In *AAAI*. 9630–9638.
[64] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *NeurIPS*. 2990–2999.
[65] James Seale Smith, Junjiao Tian, Shaunak Halbe, Yen-Chang Hsu, and Zsolt Kira. 2023. A closer look at rehearsal-free continual learning. In *CVPR*. 2409–2419.
[66] Shagun Sodhani, Sarath Chandar, and Yoshua Bengio. 2020. Toward training recurrent neural networks for lifelong learning. *Neural computation* 32, 1 (2020), 1–35.
[67] Le Sun, Mingyang Zhang, Benyou Wang, and Prayag Tiwari. 2023. Few-Shot Class-Incremental Learning for Medical Time Series Classification. *IEEE Journal of Biomedical and Health Informatics* (2023).
[68] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2016. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022* (2016).
[69] Gido M van de Ven and Andreas S Tolias. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734* (2019).
[70] Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. 2022. Three types of incremental learning. *Nature Machine Intelligence* (2022), 1–13.
[71] Jeffrey S Vitter. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)* 11, 1 (1985), 37–57.
[72] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2023. A comprehensive survey of continual learning: Theory, method and application. *arXiv preprint arXiv:2302.00487* (2023).
[73] Shaokun Wang, Weiwei Shi, Songlin Dong, Xinyuan Gao, Xiang Song, and Yihong Gong. 2023. Semantic Knowledge Guided Class-Incremental Learning. *IEEE Transactions on Circuits and Systems for Video Technology* 33, 10 (2023), 5921–5931.
[74] Shaokun Wang, Weiwei Shi, Yuhang He, Yifan Yu, and Yihong Gong. 2023. Non-Exemplar Class-Incremental Learning via Adaptive Old Class Reconstruction. In *Proceedings of the 31st ACM International Conference on Multimedia*. New York, NY, USA, 4524–4534.
[75] Zhepei Wang, Cem Subakan, Efthymios Tzinis, Paris Smaragdis, and Laurent Charlin. 2019. Continual learning of new sound classes using generative replay. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 308–312.
[76] Zhepei Wang, Cem Subakan, Efthymios Tzinis, Paris Smaragdis, and Laurent Charlin. 2019. Continual Learning of New Sound Classes Using Generative Replay. In *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. 308–312. https://doi.org/10.1109/WASPAA.2019.8937236
[77] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. 2022. Learning to prompt for continual learning. In *CVPR*. 139–149.
[78] Gary M Weiss. 2019. Wisdm smartphone and smartwatch activity and biometrics dataset. *UCI Machine Learning Repository: WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set* 7 (2019), 133190–133202.
[79] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. 2022. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *The eleventh international conference on learning representations*.
[80] Weiran Yao, Yuewen Sun, Alex Ho, Changyin Sun, and Kun Zhang. 2022. Learning Temporally Causal Latent Processes from General Temporal Data. In *International Conference on Learning Representations*.
[81] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *CVPR*. 8715–8724.
[82] Shao-Yu Yin, Yu Huang, Tien-Yu Chang, Shih-Fang Chang, and Vincent S Tseng. 2023. Continual learning with attentive recurrent neural networks for temporal data classification. *Neural Networks* 158 (2023), 171–187.
[83] Longhui Yu, Tianyang Hu, Lanqing HONG, Zhen Liu, Adrian Weller, and Weiyang Liu. 2023. Continual Learning by Modeling Intra-Class Variation. *Transactions on Machine Learning Research* (2023).
[84] Yunfei Zhang, Xiaoyang Huo, Tianyi Chen, Si Wu, and Hau San Wong. 2023. Exploring Intra-class Variation Factors with Learnable Cluster Prompts for Semi-supervised Image Synthesis. In *CVPR*. 7392–7401.
[85] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. 2023. Deep class-incremental learning: A survey. *arXiv preprint arXiv:2302.03648* (2023).
[86] Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. 2023. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems* 36 (2023), 43322–43355.
[87] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. 2021. Prototype Augmentation and Self-Supervision for Incremental Learning. In *CVPR*. 5871–5880.

Algorithm 1: Learning incremental task t for ER-based methods

Input : Memory $\mathcal{M}$, Parameters θ^*_{t-1}, Training set $\mathcal{D}^t$
Transfer model parameters: $\theta_t \leftarrow \theta^*_{t-1}$
while not converged **do**
  for $B \sim \mathcal{D}^t$ **do** // one epoch
    $B_{\mathcal{M}} \leftarrow MemoryRetrieval(\mathcal{M})$
    $\theta_t \leftarrow \mathrm{SGD}(B \cup B_{\mathcal{M}}, \theta_t)$
Obtain optimal parameter θ^*_t
for $B \sim \mathcal{D}^t$ **do** // additional pass
  $\mathcal{M} \leftarrow MemoryUpdate(B, \mathcal{M}, \theta^*_t)$
return θ^*_t, $\mathcal{M}$

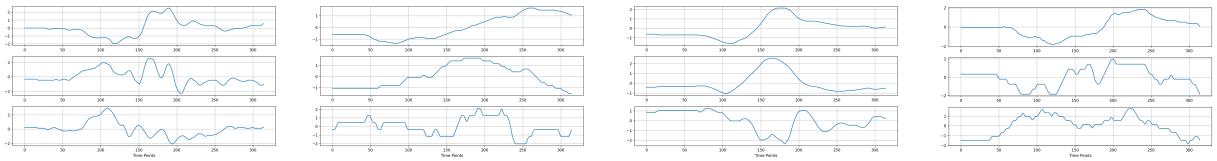
A DETAILS OF THE FRAMEWORK

A.1 Details of algorithms

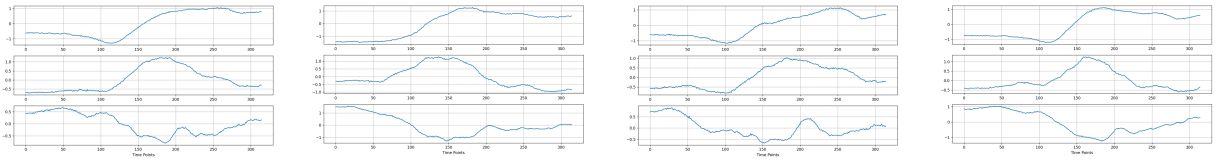
Here we introduce some details of ER, GR and TS-specific methods. Other generic methods follow their formal implementations [8]. (1) **ER**: The ER-based learning pipeline is outlined in Algorithm 1. Baisc ER [10, 59] uses Reservoir Sampling [71] as *MemoryUpdate* policy and random selection as *MemoryRetrieval* policy. (2) **GR**: We follow [64, 70] to train a generator g_ϕ along with the learner f_θ. Before task t, copies of $f_{\theta^*_{t-1}}$ and $g_{\phi^*_{t-1}}$ are saved. In each step, a batch of pseudo samples are generated and assigned labels by $f_{\theta^*_{t-1}}$. We train g_{ϕ_t} after training of f_{θ_t} is complete. Once the task is over, the saved copies are updated. (3) **DT2W**: DT^{2}W [54] employs Soft-DTW [15] to distill intermediate feature maps. In addition to this temporal KD loss, it integrates LwF to regularize the whole model. To calibrate the bias in the classification head, the method incorporates feature replay via Prototype Augmentation [74, 87]. (4) **CLOPS**: Original CLOPS [35] tailors both *MemoryUpdate* and *MemoryRetrieval* policies. However, we observe that their *MemoryRetrieval* policy encounters a limitation as replayed samples are updated after multiple epochs, potentially leading to overfitting and diminishing the effectiveness of replay. In response to this, we opt to employ random selection for memory retrieval in our experiments. (5) **FastICARL**: It is a fast variant of iCaRL, replacing herding with a *MemoryUpdate* strategy based on KNN and a max-heap. We omit the quantization process which compresses memory samples for a fair comparison.

A.2 More implementation details

For early stopping, we set a patience of 20 for ER-based and GR-based methods, and a patience of 5 for other methods. During the learning of each task, a separate validation set is split from the training data of that task, with a split ratio 1:9. The earlystopping is triggered based on the validation loss calculated on that split validation set. For hyperparameters, we tuned the learning rate and batchsize and consistently found that the best configuration was 0.001 and 64. Following [51], the choice of the learning rate scheduler is set as an hyperparameter, selecting from the following strategies: (1) step10, (2) step15 and (3) OneCycleLR [**?**]. The first two scheduler decay the learning rate by 0.1 after 10 epochs and 15 epochs, respectively. For UCI-HAR and UWave, we follow [54] to set the dropout rate to 0, while for other datasets, it is set at 0.3. For methods with replayed samples, we set the batchsize to 32 for both B and $B_{\mathcal{M}}$, ensuring the total number of samples used per step is



(a) Raw samples



(b) Generated samples

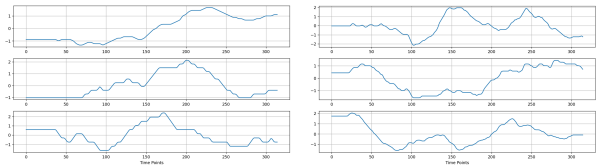
Figure 6: Comparative analysis of raw and GR-generated samples within a single UWave class. While the raw samples in 6(a) display a diversity of waveforms, the generated samples in 6(b) tend to exhibit a homogeneous pattern.

the same across all the methods. For each run, we tune the best hyperparameters by running 2 validation runs on validation tasks. Detailed hyperparameter grids can be found in the code base.

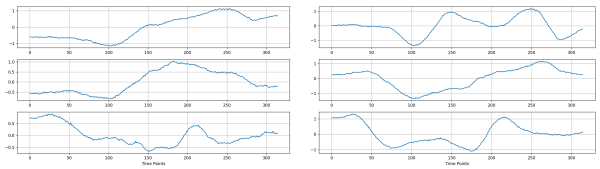
B FURTHER EXPERIMENT RESULTS

B.1 Visualization of synthesized samples of GR

To reflect the effect of generation, we show some raw and generated samples from the UWave dataset. We select this dataset since the its samples only have 3 variables, which are easy to visualize and compare. Figure 7 displays examples of raw and generated samples from two UWave classes. Specifically, the generator model is the optimized version right after learning the first task, i.e. $g_{\phi_1^*}$. The generator is used to synthesize the two classes in the first task. We note that the generated samples are unlabelled and their class labels are determined by passing them to the classification model $f_{\theta_1^*}$. We can see that the generated samples are similar to the shown original ones, demonstrating that GR is capable of synthesizing some real patterns in simple TS data.



(a) Raw samples



(b) Generated samples

Figure 7: Comparison of raw and GR-synthesized UWave samples across 2 classes. Left and right columns represent distinct classes respectively.

Apart from the above, we also investigate the diversity of the generated samples. Similarly, we focus on the UWave dataset and use the generator model after learning the first task. Unlike previous comparisons, we only examine the raw and generated samples from a single class. Figure 6 selectively depicts 8 samples from the single UWave class, comprising 4 raw and 4 generated samples. We can observe that raw samples appear to exhibit several different patterns, demonstrating the diversity present in the original data. In contrast, generated sample prone to exhibit a few similar patterns. This indicates that a potential challenge for using GR in TSCIL is to ensure the diversity of generated data.

B.2 How do intra-class variations affect TSCIL?

To further understand the role of intra-class variations caused by subjects, we consider a simple scenario that using ER to learn a sequence of 2 tasks. Here we assume that both tasks are sourced from the same group subjects $S = \{s_1, \ldots, s_k\}$, with a prior distribution as $p(S) = \sum_{s \in S} p(S = s)$. The input distribution is conditioned on subject s and can be represented as $p(X) = \sum_{s \in S} p(X|S = s)p(S = s)$. For incrementally learning the two tasks with distributions $p(\mathcal{X}_1, \mathcal{Y}_1)$ and $p(\mathcal{X}_2, \mathcal{Y}_2)$, the learning objective can be formulated as:

$$\max \sum_{s \in S} p_1(s) \sum_{(\mathbf{x}_1, y_1) \in \mathcal{D}_1} \log p(y_1|\mathbf{x}_1, s) + \sum_{s \in S} p_2(s) \sum_{(\mathbf{x}_2, y_2) \in \mathcal{D}_2} \log p(y_2|\mathbf{x}_2, s) \tag{2}$$

where $p_1(s)$ and $p_2(s)$ represent the distribution of subjects in task 1 and task 2, respectively. When learning the second task, the second term is calculated on incoming training set thus it is clear that $p_2(s) = p(s)$. However, the first term is computed based on replaying memory samples. But typical memory management strategies tend to overlook the implicit influence of subject s and omit to maintain $p_1(s) = p(s)$, making the replayed data deviated from the original one. This problem can be extended to general cases with more tasks, showing the influence of variations caused by different subjects.