Dynamic Channel Charting: Integrating Online Sample Selection with Continual Learning for Streaming CSI Data

Yamil Vindas, Maxime Guillaud

Dynamic Channel Charting: Integrating Online Sample Selection with Continual Learning for Streaming CSI Data

Yamil Vindas, Maxime Guillaud
Inria, INSA Lyon, CITI Laboratory, UR3720, F-69621 Villeurbanne, France

***Abstract*—Wireless channel charting consists in the application of dimensionality reduction methods to the channel state information (CSI) collected during the operation of wireless communication systems. Due to hardware limitations, it is desirable to limit the amount of information stored for that purpose; hence, this work considers online sample selection to perform channel charting from streaming CSI data. We specifically focus on the case where dimensionality reduction is performed via contrastive learning using sample triplets, and study several suitable sample selection strategies. The proposed approaches are numerically evaluated and compared using measured data.**

I. Introduction

Channel charting (CC) is an unsupervised learning technique that utilizes channel state information (CSI) to construct a low-dimensional representation of the radio propagation environment [1]. Its applications include predictive radio resource management and beam management, proximity detection, context awareness for digital twin applications, and improving localization [2].

A. Related work

1) Channel charting: Early methods for CC relied on non-parametric dimensionality reduction (DR) techniques, such as Sammon's mapping and principal component analysis (PCA) [1]. Recent advances leverage parametric deep learning models, including fully-connected (FC) autoencoders [1], Siamese networks [3] and Triplet networks [4]. These deep learning approaches offer improved handling of the out-of-sample problem, making them more effective for real-world applications.

Moreover, many applications of CC require applying DR to a large number of CSI samples, which are measured at high sampling rates (typically around 100 samples per second) by each base station (BS). BSs often have limited computational and storage resources, especially for auxiliary tasks like CC, as their primary function is to manage telecommunications. Consequently, it is not feasible to store all the generated CSI samples. Furthermore, since it is desirable to apply CC over data collected over long durations (potentially weeks or months), it is reasonable to assume that the total amount of CSI that will be acquired is a priori unbounded. Thus, it is beneficial to consider online methods that can generate partial, incremental solutions that can be refined as additional data becomes available. A promising framework to address this is continual learning. For instance, [5] introduced an online replay-based strategy to retain a limited set of CSI samples for performing CC.

2) Continual learning (CL): CL is a recent field of machine learning research focused on developing methods that enable models to continuously learn and adapt to new data without forgetting previously acquired knowledge, a phenomenon know as catastrophic forgetting. Various families of methods exist in continual learning [6]; in this work, we focus on replay-based strategies, which involve maintaining a small memory of real or synthetic samples, which aids models in retaining information that might otherwise be lost over time. Replay-based strategies are well-suited for online settings with high-rate streaming data—provided that the memory update mechanism is efficient. Moreover, these methods are inherently compatible with unsupervised online continual learning, as they share a close relationship with core-set approaches, which have proven effective in unsupervised scenarios [7].

Core-set approaches, although initially not designed for deep learning, posed challenges when applied to such models because of their complexity and tendency towards poor generalization [8]. Nevertheless, recent years have seen substantial progress in adapting these techniques to deep learning contexts [9], [10]. Recent methods have integrated gradient-matching techniques with core-sets, enabling replay-based continual learning that is suitable for online settings [9], [10]. The central idea here is to preserve the gradient information of the entire dataset through the core-set.

Additionally, while the term "core-set" is often used to describe a long-term memory in replay-based approaches for continual learning, these methods do not necessarily align with the strict mathematical definitions of core-sets as outlined in [7]. Thus, the focus shifts towards long-term memory construction rather than strictly adhering to core-set definitions. Different approaches either generate synthetic samples or store other useful information (e.g., model outputs, gradients) to represent previously seen samples [11], [12]. Another class of methods employs loss or gradient information [13], [14], [15], as well as other metrics (e.g., sample similarity, entropy) [5], [16] to select samples that maximize the information content

This work was supported by ANR (grant ANR-23-CHR4-0001-01) under the CHIST-ERA project CHASER (CHIST-ERA-22-WAI-01), and by Horizon Europe SNS project INSTINCT (grant 101139161). It was performed using HPC resources from GENCI-IDRIS (Grant 2023-AD010614930).

of the memory.

The primary limitations of previous studies are that they are often tailored to supervised learning contexts or fail to effectively address online scenarios, where data arrives at high rates. In such cases, it is crucial to rapidly update the model and memory to prevent significant data loss and ensure efficient model training.

B. Summary of the Contributions

- We introduce a dual memory architecture based on a long-term memory acting as the core-set reservoir, and a short-term buffer which can store incoming samples during model updates and memory updates;
- We introduce several memory update strategies suitable for the contrastive learning goal of triplet-based CC;
- We benchmark the proposed approaches using real-world data from the DICHASUS measurement platform [17].

II. Problem Definition

A. CC through Contrastive Learning

Assume that we observe a stochastic process generating data over time, i.e. for all $t \geq 0$, $\mathbf{x}_t \in \mathbb{R}^d$ is a sample of dimension d acquired a time t. For simplicity we discretize time and index it by $t \in \mathbb{N}$, denoting by $t_{\mathbf{x}}$ the timestamp of sample $\mathbf{x}$. Our main goal is to train a neural network model f_θ with parameters θ to perform CC over an undefined period of time, while avoiding as much as possible catastrophic forgetting.

In this work, we specifically focus on CC via triplet-based contrastive learning. In this approach, the CC problem is subsumed by a metric learning approach which relies on the timestamps associated with each sample. Specifically, we seek to train a model denoted by h to learn a distance. This distance, defined as $\mathrm{d}_h : (\mathbf{x}, \mathbf{y}) \mapsto ||h(\mathbf{x}) - h(\mathbf{y})||_2$, is learned through triplets of (*anchor, positive, negative*) samples $(\mathbf{a}, \mathbf{p}, \mathbf{n}) \in \mathbb{R}^{d\times 3}$ for which we know that d_h should fulfill $\mathrm{d}_h(\mathbf{a}, \mathbf{p}) < \mathrm{d}_h(\mathbf{a}, \mathbf{n})$. It has been shown in [4] that this distance learning problem is a good proxy for the CC problem. In practice, such triplets can be easily derived from the timestamp information by selecting samples sufficiently close in time (say, for which $|t_{\mathbf{a}} - t_{\mathbf{p}}| \leq T_c$, where T_c is below the expected channel coherence time) for the positive pair $(\mathbf{a}, \mathbf{p})$; and selecting a third sample at random for which $|t_{\mathbf{a}} - t_{\mathbf{n}}| > T_c$ will yield the (likely) negative example $\mathbf{n}$.

For a given dataset $\mathcal{A} \subset \mathbb{R}^d$ and for any anchor sample $\mathbf{a} \in \mathcal{A}$, we let $\mathcal{T}_{\mathbf{a}}(\mathcal{A})$ denote the set of positive and negative sample pairs from $\mathcal{A}$ associated to $\mathbf{a}$, which is defined according to their respective timestamps: $\forall \mathbf{a} \in \mathcal{A}$,

$$\mathcal{T}_{\mathbf{a}}(\mathcal{A}) = \{(\mathbf{p}, \mathbf{n}) \in \mathcal{A} \times \mathcal{A} \text{ s.t. } |t_{\mathbf{a}} - t_{\mathbf{p}}| \leq T_c < |t_{\mathbf{a}} - t_{\mathbf{n}}|\} \tag{1}$$

The triplet loss associated with an anchor sample $\mathbf{a} \in \mathcal{A}$, relative to a model h, is defined as follows:

$$\mathcal{L}_T(\mathbf{a}, \mathcal{T}_{\mathbf{a}}(\mathcal{A}); h) = \frac{1}{|\mathcal{T}_{\mathbf{a}}(\mathcal{A})|} \cdot \sum_{(\mathbf{p},\mathbf{n})\in\mathcal{T}_{\mathbf{a}}(\mathcal{A})} \ell_T(\mathbf{a}, \mathbf{p}, \mathbf{n}; h) \tag{2}$$

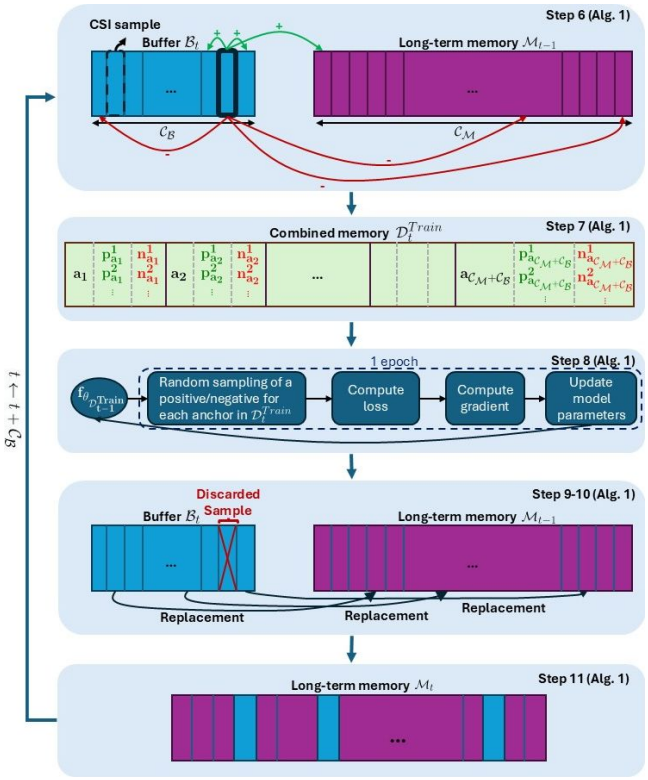


Fig. 1: **Problem Summary.** Our framework involves steps to periodically update long-term memory ($\mathcal{M}_{t-1}$ at time t) and train the model with new data from the buffer $\mathcal{B}_t$. At time t, the model is trained with the dataset $\mathcal{D}_t^{\text{Train}} = \mathcal{B}_t \cup \mathcal{M}_{t-1}$. Each sample in $\mathcal{D}_t^{\text{Train}}$ serves as an anchor for triplet learning, with positive and negative examples drawn from $\mathcal{D}_t^{\text{Train}}$ (indicated by the red and green arrows in the first block). Each block corresponds to a line in our proposed online training Alg. 1.

where $\ell_T(\mathbf{a}, \mathbf{p}, \mathbf{n}; h) = \max(0, \mathrm{d}_h(\mathbf{a}, \mathbf{p}) - \mathrm{d}_h(\mathbf{a}, \mathbf{n}) + M)$ and M is a margin hyperparameter that defines the minimum required distance between the anchor/positive and anchor/negative pairs.

B. Online Learning

We suppose that we have a long-term memory $\mathcal{M}$ with a limited storage capacity, allowing it to hold up to $C_{\mathcal{M}}$ samples. For initialization, incoming samples are added to $\mathcal{M}$ sequentially until it reaches full capacity. We also suppose that we have a buffer $\mathcal{B}$ with capacity $C_{\mathcal{B}}$ which temporarily stores at most $C_{\mathcal{B}}$ of the latest received samples (see Figure 1). We assume that we can process all samples in the buffer before they are replaced by new incoming samples (no buffer overrun). Additionally, samples are processed in batches, meaning that at a given time t, we process all the samples in the buffer $\mathcal{B}_t$ to determine whether they should be added to the long-term memory $\mathcal{M}_t$. After processing, all samples in $\mathcal{B}_t$ are replaced by the new set of $C_{\mathcal{B}}$ samples.

We denote as $\mathcal{M}_t$ the content of the memory at time t, $\mathcal{B}_t$ the content of the buffer at time t, and $f_{\mathcal{M}_{\theta_t}}$ the model trained with $\mathcal{M}_t$. We suppose that $f_{\theta_{\mathcal{M}_{t-1}}}$ is retrained each time the buffer memory is updated, using $\mathcal{D}_t^{\text{Train}} = \mathcal{B}_t \cup \mathcal{M}_{t-1}$.

Similar to the continual learning set-up in [18] and other memory-based approaches, at each time step t, we aim to find a memory $\mathcal{M}_t$ that enables training the model f_θ such that its test performance closely approximates that of training with the full dataset at time step t, $\mathcal{D}_t = \{\mathbf{x}_0, \dots, \mathbf{x}_t\}$, which consists of all samples received up to time t, i.e.

$$\mathop{\mathbb{E}}_{(\mathbf{a},\mathbf{p},\mathbf{n})\sim\mathcal{P}_\mathrm{T}}[\ell_T(\mathbf{a},\mathbf{p},\mathbf{n};f_{\theta_{\mathcal{D}_t}})] \approx \mathop{\mathbb{E}}_{(\mathbf{a},\mathbf{p},\mathbf{n})\sim\mathcal{P}_\mathrm{T}}[\ell_T(\mathbf{a},\mathbf{p},\mathbf{n};f_{\theta_{\mathcal{M}_t}})] \tag{3}$$

with $\theta_{\mathcal{D}_t} = \arg\min_\theta \frac{1}{|\mathcal{D}_t|} \sum_{\mathbf{x}\in\mathcal{D}_t} \mathcal{L}_T(\mathbf{x}, \mathcal{T}_\mathbf{x}(\mathcal{D}_t); f_\theta)$ and $\theta_{\mathcal{M}_t} = \arg\min_\theta \frac{1}{\mathcal{C}_\mathcal{M}} \sum_{\mathbf{x}\in\mathcal{M}_t} \mathcal{L}_T(\mathbf{x}, \mathcal{T}_\mathbf{x}(\mathcal{M}_t); f_\theta)$ where $\mathcal{P}_\mathrm{T}$ is the distribution of triplets of the test dataset, $\mathbf{p}$ and $\mathbf{n}$ are positive and negative samples associated to anchor $\mathbf{a}$, $f_{\theta_{\mathcal{D}_t}}$ is the model trained using $\mathcal{D}_t$, and $f_{\theta_{\mathcal{M}_t}}$ is the model trained using $\mathcal{M}_t$.

Note that since the positive and negative samples associated with an anchor sample depend on their temporal distance (the difference between timestamps), and since samples are added and removed over time from the long-term memory $\mathcal{M}$, the number of positive samples associated with anchors in $\mathcal{M}$ tends to decrease and may potentially reach zero. In this case, the anchor sample in memory becomes unusable for contrastive training.

To ensure that, for each anchor $\mathbf{a} \in \mathcal{M}_t$, we have $\mathcal{T}_\mathbf{a}(\mathcal{M}_t) \neq \emptyset$ at every time step t, when updating $\mathcal{M}_{t-1}$ we compute $\mathcal{T}_\mathbf{a}(\mathcal{B}_t \cup \mathcal{M}_{t-1})$ and randomly selecting one positive sample to store in $\mathcal{M}_t$ alongside the anchor. Since each anchor $\mathbf{a}$ is added to memory directly from the buffer, there is always at least one positive sample available in the buffer at that time. This is preferable to directly storing triplets in $\mathcal{M}$ as it allows for different positives for each anchor whenever possible.

III. Considered solutions

A. State-of-the-Art Methods

1) Random selection: A straightforward yet naive approach to online sample selection involves randomly determining whether to retain each incoming sample from the buffer in memory according to a fixed (non data-dependent) probability. If the sample is retained, another sample in memory is randomly chosen for removal to make space. Samples from the buffer are processed in order of arrival; therefore, the long-term memory update is defined as follows: for all $t > \mathcal{C}_\mathcal{M}$ and $\tau \in [t - \mathcal{C}_\mathcal{B}, t]$:

$$\mathcal{M}_\tau = \begin{cases} (\mathcal{M}_{\tau-1} \setminus \{\mathbf{x}_r\}) \cup \{\mathbf{x}_\tau\} \text{ with probability } p \\ \mathcal{M}_{\tau-1} \text{ with probability } 1-p \end{cases} \tag{4}$$

where $\mathbf{x}_r$ is a sample to be removed chosen uniformly at random from the memory $\mathcal{M}_{\tau-1}$.

This approach suffers from catastrophic forgetting as, for all $t \geq \mathcal{C}_\mathcal{M}$ and $\tau \in [\mathcal{C}_\mathcal{M}, t]$ the probability of sample $\mathbf{x}_\tau$ to be in the long term memory $\mathcal{M}_t$ at time step t is given by

$$\mathbb{P}(\mathbf{x}_\tau \in \mathcal{M}_t) = p \cdot \left(1 - \frac{1}{\mathcal{C}_\mathcal{M}}\right)^{t-\tau}. \tag{5}$$

We observe that for t fixed, $\mathbb{P}(\mathbf{x}_\tau \in \mathcal{M}_t)$ decreases for smaller values of τ; this indicates that earlier samples are less likely to be included in $\mathcal{M}_t$.

2) Reservoir sampling (RS): Introduced in [19], this memory update strategy operates on individual samples, without the concept of a buffer (i.e., it relies solely on the long-term memory). Given a timestamp t, the long-term memory update rule is defined as follows: for all $t > \mathcal{C}_\mathcal{M}$,

$$\mathcal{M}_t = \begin{cases} (\mathcal{M}_{t-1} \setminus \{\mathbf{x}_r\}) \cup \{\mathbf{x}_t\} \text{ with probability } \frac{\mathcal{C}_\mathcal{M}}{t} \\ \mathcal{M}_{t-1} \text{ else} \end{cases} \tag{6}$$

where r is an integer sampled uniformly at random from $[1, t]$. It corresponds to uniformly sampling $\mathcal{C}_\mathcal{M}$ instances from the set $\mathcal{D}_t$, which denotes all samples observed up to time t. Therefore, at time t, the probability of sample $\mathbf{x}_\tau$ with $\tau \leq t$ to be in the long term memory $\mathcal{M}_t$ is given by:

$$\mathbb{P}(\mathbf{x}_\tau \in \mathcal{M}_t) = \frac{\mathcal{C}_\mathcal{M}}{t}. \tag{7}$$

Thus RS enables an online implementation of a uniform random sample selection policy.

3) SimS: This approach was proposed in a more advanced form for supervised learning in [14] and in a simpler form for unsupervised learning in [5]. The main objective is to minimize the similarity between samples in $\mathcal{M}_t$ to encourage diversity within the memory set. Let D denote a distance over $\mathbb{R}^d$; SimS aims to maximize the minimum distance within $\mathcal{M}_t$:

$$\max_{\mathcal{M}_t \subset \mathcal{D}_t} \left(\min_{(\mathbf{u},\mathbf{v})\in\mathcal{M}_t} \mathrm{D}(\mathbf{u},\mathbf{v}) \right) \quad \text{s.t.} \quad |\mathcal{M}_t| = \mathcal{C}_\mathcal{M}. \tag{8}$$

Solving (8) in an online fashion is not possible because we do not have access to all samples simultaneously. [5] introduced the following greedy algorithm to find an approximate solution through the following memory update rule: for all $t > \mathcal{C}_\mathcal{M}$ and $\tau \in [t - \mathcal{C}_\mathcal{B}, t]$, let $\mathbf{u} = \arg\min_{\mathbf{x}\in\mathcal{M}_{\tau-1}} \mathrm{D}(\mathbf{x}_\tau, \mathbf{x})$ denote the nearest neighbor in memory $\mathcal{M}_{\tau-1}$ to the new sample $\mathbf{x}_\tau$. The proposed memory update rule is then defined as

$$\mathcal{M}_\tau = \begin{cases} (\mathcal{M}_{\tau-1} \setminus \{\mathbf{x_r}\}) \cup \{\mathbf{x}_\tau\} \\ \qquad \text{if } \mathrm{D}(\mathbf{x}_\tau, \mathbf{u}) > \mathrm{D}(\mathbf{v}, \mathbf{w}), \\ \mathcal{M}_{\tau-1} \text{ else} \end{cases} \tag{9}$$

where $(\mathbf{v}, \mathbf{w}) = \arg\min_{(\mathbf{x},\mathbf{y})\in\mathcal{M}_{\tau-1}^2} \mathrm{D}(\mathbf{x}, \mathbf{y})$, $\mathbf{x_r} = \mathbf{v}$ with probability p and $\mathbf{x_r} = \mathbf{w}$ with probability $1-p$. When working with a buffer and long term memory, each update consists in solving Eq. (8) by replacing $\mathcal{D}_t$ with $\mathcal{B}_t \cup \mathcal{M}_{t-1}$, processing samples in the buffer sequentially in order of arrival.

We introduce a more robust replacement strategy consisting in adjusting the selection criteria for samples to be removed from memory. Our idea involves examining the second closest neighbors of both $\mathbf{v}$ and $\mathbf{w}$, and prioritizing the removal of the sample whose closest second neighbor is the closest overall. We refer to this method as SimS-SCN, where SCN stands for "Second Closest Neighbor."

B. Proposed online model-based dataset condensation

We now introduce three novel model-based methods for performing unsupervised online dataset condensation to dynamically select samples from the buffer $\mathcal{B}_t$ for storage in the long-term memory $\mathcal{M}_t$.

1) Max-Min Point Spacing (MMPS): This approach is similar to the one from [5], with the difference that we use the learned distance $\mathrm{d}_{f_{\theta_{\mathcal{M}_{t-1}}}}$ in lieu of the sample-space distance D in eq. (8) to update the long-term memory from $\mathcal{M}_{t-1}$ to $\mathcal{M}_t$. This yields the following optimization problem:

$$\max_{\mathcal{M}_t \subset \mathcal{D}_t} \left(\min_{(\mathbf{u},\mathbf{v}) \in \mathcal{M}_t} \mathrm{d}_{f_{\theta_{\mathcal{M}_{t-1}}}}(\mathbf{u},\mathbf{v}) \right) \quad \text{s.t. } |\mathcal{M}_t| = \mathcal{C}_{\mathcal{M}} \tag{10}$$

When working with a buffer and long-term memory, an iterative approach to solve this problem consists in solving Eq. (10) at each time step by replacing $\mathcal{D}_t$ with $\mathcal{B}_t \cup \mathcal{M}_{t-1}$. The memory update rule can be defined as follows, for all $t > \mathcal{C}_{\mathcal{M}}$ and $\tau \in [t - \mathcal{C}_{\mathcal{B}}, t]$:

$$\mathcal{M}_\tau = \begin{cases} (\mathcal{M}_{\tau-1} \setminus \{\mathbf{x_r}\}) \cup \{\mathbf{x}_\tau\} \\ \qquad \text{if } \mathrm{d}_{f_{\theta_{\mathcal{M}_{\tau-1}}}}(\mathbf{x}_\tau, \mathbf{u}) > \mathrm{d}_{f_{\theta_{\mathcal{M}_{\tau-1}}}}(\mathbf{v}, \mathbf{w}), \\ \mathcal{M}_{\tau-1} \text{ else} \end{cases} \tag{11}$$

where $\mathbf{u} = \arg\min_{\mathbf{x} \in \mathcal{M}_{\tau-1}} \mathrm{d}_{f_{\theta_{\mathcal{M}_{\tau-1}}}}(\mathbf{x}_\tau, \mathbf{x})$, $(\mathbf{v}, \mathbf{w}) = \arg\min_{(\mathbf{x},\mathbf{y}) \in \mathcal{M}_{\tau-1}^2} \mathrm{d}_{f_{\theta_{\mathcal{M}_{\tau-1}}}}(\mathbf{x}, \mathbf{y})$, and $\mathbf{x}_r = \mathbf{v}$ if the second closest neighbor to $\mathbf{v}$ is closer to $\mathbf{v}$ than the second closest neigbor of $\mathbf{w}$ is to $\mathbf{w}$, otherwise $\mathbf{x}_r = \mathbf{w}$.

Note that MMPS significantly differs from [5], [16] in that:

- It operates directly within the reduced embedding space, akin to the approach from [18] for supervised learning;
- It utilizes a contrastive/triplet learning framework;
- It employs a self-supervised, online CL methodology.

2) Loss-Driven Ranking Sample Selection (LDRSS): This approach is comparable to the offline supervised continual learning method discussed in [13]. However, our method introduces a loss function-based unsupervised *online* CL framework. The primary objective is to enhance the difficulty of the samples stored in memory by maximizing their loss function, as samples with higher loss tend to be more informative to the model that those with smaller loss. This strategy promotes diversity and complexity within the memory set.

LDRSS aims at solving the following problem at time t:

$$\max_{\mathcal{M}_t \subset \mathcal{D}_t} \left(\sum_{\mathbf{x} \in \mathcal{M}_t} \mathcal{L}_T(\mathbf{x}, \mathcal{T}_{\mathbf{x}}(\mathcal{M}_t); f_{\theta_{\mathcal{M}_{t-1}}}) \right) \text{ s.t. } |\mathcal{M}_t| = \mathcal{C}_{\mathcal{M}} \tag{12}$$

Again, solving this optimization problem is not practical if we do not have access to all samples simultaneously, so we propose to find an approximate solution by iteratively solving (12) by replacing $\mathcal{D}_t$ with $\mathcal{B}_t \cup \mathcal{M}_{t-1}$.

Memory $\mathcal{M}_{t-1}$ is updated using the samples in the buffer $\mathcal{B}_t$ one by one in order of arrival, according to the following memory update rule[1]: for all $t > \mathcal{C}_{\mathcal{M}}$ and $\tau \in [t - \mathcal{C}_{\mathcal{B}}, t]$,

$$\mathcal{M}_\tau = \begin{cases} (\mathcal{M}_{\tau-1} \setminus \{\mathbf{x_r}\}) \cup \{\mathbf{x}_\tau\} \text{ if } \mathcal{L}_T(\mathbf{x}_\tau, \mathcal{T}_{\mathbf{x}_\tau}(\mathcal{B}_t \cup \\ \mathcal{M}_{\tau-1}); f_{\theta_{\mathcal{M}_{\tau-1}}}) > \mathcal{L}_T(\mathbf{x}_r, \mathcal{T}_{\mathbf{x}_r}(\mathcal{M}_{\tau-1}); f_{\theta_{\mathcal{M}_{\tau-1}}}) \\ \mathcal{M}_{\tau-1} \text{ else} \end{cases} \tag{13}$$

[1]It is important to note that updating the memory using sample losses incurs minimal computational overhead since we can leverage the fact that those losses are already computed for the purpose of model training.

where $\mathbf{x_r} = \arg\min_{\mathbf{u} \in \mathcal{M}_{\tau-1}} \mathcal{L}_T(\mathbf{u}, \mathcal{T}_{\mathbf{u}}(\mathcal{M}_{\tau-1}); f_{\theta_{\mathcal{M}_{\tau-1}}})$.

3) Gradient-Driven Ranking Sample Selection (GDRSS): This approach is comparable to the online CL method from [14]. However, our method employs a direct gradient-based scoring mechanism to select samples from the buffer for long-term memory retention, since samples with larger gradients tend to be more informative for model learning. The primary goal is to retain the samples that contribute most significantly to reaching a local minimum. GDRSS aims at solving the following optimization problem at time t:

$$\max_{\mathcal{M}_t \subset \mathcal{D}_t} \sum_{\mathbf{x} \in \mathcal{M}_t} \mathcal{N}\left(\nabla_\theta \mathcal{L}_T(\mathbf{x}, \mathcal{T}_{\mathbf{x}}(\mathcal{M}_t); f_{\theta_{\mathcal{M}_{t-1}}}) \right) \quad \text{s.t. } |\mathcal{M}_t| = \mathcal{C}_{\mathcal{M}} \tag{14}$$

where $\nabla_\theta \mathcal{L}_T$ corresponds to the gradient of the loss function with respect to the parameters θ of a function f_θ, and $\mathcal{N}$ is a function reducing the gradient into a scalar value (for instance a norm), defined as follows: for all $\mathbf{u} \in \mathcal{A}$,

$$\mathcal{N}(\nabla_\theta \mathcal{L}_T(\mathbf{u}, \mathcal{T}_{\mathbf{u}}(\mathcal{A}); f_\theta)) = \frac{1}{L} \sum_{l=1}^{L} \frac{1}{n_l} ||\nabla_{\theta^l} \mathcal{L}_T(\mathbf{u}, \mathcal{T}_{\mathbf{u}}(\mathcal{A}); f_\theta))||_1$$

where L denotes the number of layers of the model f_θ, θ^l the parameters of layer $l \in [1, L]$, and n_l is the number of parameters in layer l.

For the online setting, a greedy algorithm to find an approximate solution can be obtained by solving Eq. (14) sequentially and replacing $\mathcal{D}_t$ with $\mathcal{B}_t \cup \mathcal{M}_{t-1}$. The memory update rule can be defined as follows, for all $t > \mathcal{C}_{\mathcal{M}}$ and $\tau \in [t - \mathcal{C}_{\mathcal{B}}, t]$:

$$\mathcal{M}_\tau = \begin{cases} (\mathcal{M}_{\tau-1} \setminus \{\mathbf{x_r}\}) \cup \{\mathbf{x}_\tau\} \text{ if } \tilde{\mathcal{G}}_T(\mathbf{x}_\tau, \mathcal{T}_{\mathbf{x}_\tau}(\mathcal{B}_t \cup \\ \mathcal{M}_{\tau-1}); f_{\theta_{\mathcal{M}_{\tau-1}}}) > \tilde{\mathcal{G}}_T(\mathbf{x}_r, \mathcal{T}_{\mathbf{x}_r}(\mathcal{M}_{\tau-1}); f_{\theta_{\mathcal{M}_{\tau-1}}}) \\ \mathcal{M}_{\tau-1} \text{ else} \end{cases} \tag{15}$$

where $\forall \mathbf{u} \in \mathcal{A}$, $\tilde{\mathcal{G}}_T(\mathbf{u}, \mathcal{T}_{\mathbf{u}}(\mathcal{A}); f_\theta) = \mathcal{N}(\nabla_\theta \mathcal{L}_T(\mathbf{u}, \mathcal{T}_{\mathbf{u}}(\mathcal{A}); f_\theta))$, and $\mathbf{x_r} = \arg\min_{\mathbf{u} \in \mathcal{M}_{\tau-1}} \tilde{\mathcal{G}}_T(\mathbf{u}, \mathcal{T}_{\mathbf{u}}(\mathcal{M}_{\tau-1}); f_{\theta_{\mathcal{M}_{\tau-1}}})$.

C. Training strategy

By utilizing a buffer $\mathcal{B}_t$ and long-term memory $\mathcal{M}_{t-1}$, we can employ a continual learning paradigm based on replay. In this approach, each time the model is trained, it is updated using $\mathcal{D}_t^{\text{Train}}$ composed of all samples from both $\mathcal{B}_t$ and $\mathcal{M}_{t-1}$. This process involves solving

$$\min_\theta \lambda_{\mathcal{B}} \cdot \mathcal{L}_{\mathcal{B}}(f_\theta; \mathcal{B}_t) + \lambda_{\mathcal{M}} \cdot \mathcal{L}_{\mathcal{M}}(f_\theta; \mathcal{M}_{t-1}) \tag{16}$$

where $\mathcal{L}_{\mathcal{B}}(f_\theta; \mathcal{B}_t)$ is the current data loss (obtained with the buffer) and $\mathcal{L}_{\mathcal{M}}(f_\theta; \mathcal{M}_{t-1})$ is the replay loss on the long-term memory, defined as follows:

$$\mathcal{L}_{\mathcal{B}}(f_\theta; \mathcal{B}_t) = \frac{1}{\mathcal{C}_{\mathcal{B}}} \sum_{\mathbf{x} \in \mathcal{B}_t} \mathcal{L}_T(\mathbf{x}, \mathcal{T}_{\mathbf{x}}(\mathcal{B}_t \cup \mathcal{M}_{\tau-1}); f_\theta) \tag{17}$$

$$\mathcal{L}_{\mathcal{M}}(f_\theta; \mathcal{M}_t) = \frac{1}{\mathcal{C}_{\mathcal{M}}} \sum_{\mathbf{x} \in \mathcal{M}_{t-1}} \mathcal{L}_T(\mathbf{x}, \mathcal{T}_{\mathbf{x}}(\mathcal{B}_t \cup \mathcal{M}_{\tau-1}); f_\theta) \tag{18}$$

Finally, $\lambda_{\mathcal{B}}$ and $\lambda_{\mathcal{M}}$ are two hyperparameters that control the importance of recent data and long-term memory, respectively.

A generic online sample selection and replay-based continual training algorithm summarizing all previous approaches is given in Algorithm 1.

Algorithm 1: Online sample selection and training

Initialize:
  $\mathcal{M}_0 \leftarrow [\mathbf{x}_1, ..., \mathbf{x}_{\mathcal{C}_\mathcal{M}}]$;
  $\mathcal{B}_0 \leftarrow [\mathbf{x}_{\mathcal{C}_\mathcal{M}+1}, ..., \mathbf{x}_{\mathcal{C}_\mathcal{M}+\mathcal{C}_\mathcal{B}}]$;
  $f_{\theta_{\mathcal{M}_0}}$ by solving (16);
for $t = \mathcal{C}_\mathcal{B}, 2 \cdot \mathcal{C}_\mathcal{B}, ...$ **do**
  Update buffer to obtain $\mathcal{B}_t$;
  $\mathcal{D}_t^{\text{Train}} \leftarrow \mathcal{B}_t \cup \mathcal{M}_{t-1}$;
  Update the model by solving (16) to obtain $f_{\theta_{\mathcal{M}_{t-1}}}$
  for $\mathbf{x} \in \mathcal{B}_t$ **do**
    Update $\mathcal{M}_{t-1}$ using a memory update rule (one of equations (4), (6), (9), (11), (13), (15));
  end
  $\mathcal{M}_t \leftarrow \mathcal{M}_{t-\mathcal{C}_\mathcal{B}}$;
end

IV. Experimental Setup

A. Dataset

We evaluate the methods using two datasets from the DICHASUS platform of the University of Stuttgart [17], specifically `cf-02` (18,516 samples) for training, and `cf-03`, (23,478 samples) for testing [20]. Both datasets cover the same indoor industrial environment, where a robot equipped with an omnidirectional antenna traversed an L-shaped area. CSI was measured between the robot and four access points (APs), each with eight antennas. The system operates using Orthogonal Frequency Division Multiplexing with 1,024 subcarriers, over a bandwidth of 50 MHz centered at 1.272 GHz.

In order to model the kind of non-stationarity of the CSI stochastic process that can occur in real scenarios, we bias the dataset by repeating the first half (upper section) of the trajectory 11 times and the second half (lower section) once, while preserving the continuity of the trajectory. This modification resulted in a total of 111,086 samples, corresponding to approximately 1.60 hours of data collection at a sampling frequency of around 20 Hz. We denote this enhanced dataset as `cf-02-R` (`cf-02` repeated). Finally, the data underwent preprocessing according to the pipeline outlined in [1].

B. Experiments

We trained the FC network described in [21] using Algorithm 1. During each training and retraining session, we utilized 10 epochs. Throughout all simulations, models were optimized using the ADAM optimizer with a batch size of 1500, a learning rate of 1×10^{-1}, and a weight decay of 1×10^{-7}. Triplet selection was performed with $T_C = 5$ s, and ℓ_T computed with $M = 1$. Since computing $\mathcal{L}_T(\mathbf{a}, \mathcal{T}_\mathbf{a}(\mathcal{A}); h)$ and $\tilde{\mathcal{G}}_T(\mathbf{a}, \mathcal{T}_\mathbf{a}(\mathcal{A}); h)$ exactly is costly because the cardinality of the set $\mathcal{T}_\mathbf{a}(\mathcal{A})$ tends to be very large, we randomly sampled one positive/negative pairs for each anchor sample every time it is seen during training.

We fixed $\lambda_\mathcal{B} = \lambda_\mathcal{M} = 1$ in eq. (16) to compare the different approaches against each other and to an offline method where the samples within each batch were uniformly selected among all the available samples. The offline method was trained over 30 epochs, using all the `cf-02-R` training data available. This may introduce a bias toward the first half of the trajectory in each batch (as it is oversampled), leading to an oversampling in space, but not in time. Each experiment was repeated five times, and the results were averaged.

We used trustworthiness (T), continuity (C), and Kruskal stress (KS) metrics between the chart and the 2D positions where each CSI sample was acquired (in practice not available) to evaluate the preservation of the scene geometry within the ambient space. For further details, we direct the reader to [4].

V. Results and discussion

First, from Table I, we can observe that GDRSS emerges as the best overall approach among the online methods, significantly outperforming others in terms of trustworthiness and continuity, while achieving comparable Kruskal stress. Additionally, GDRSS produces stable models, as evidenced by the low standard deviation figures. Unlike RS, which treats all incoming samples equally to yield an uniform sampling in time, GDRSS chooses samples that are better suited to the model's needs, rather than relying on uniform sampling.

On the other hand, LDRSS does not achieve performance comparable to that of GDRSS, despite also being a model-based sampling selection approach that prioritizes challenging samples. This is because, although samples with higher loss values are generally more challenging to learn than those with lower loss values, a single sample does not provide sufficient information about the overall loss landscape. A sample with a high loss value might be located in a local minimum or a flat region, making it less effective for improving the model.

Furthermore, we observe that the performance of all online methods surpasses that of the offline method, which has access to the entire dataset at once. This improvement highlights a favorable side effect of our online learning framework when handling biased datasets (for instance, the `cf-02-R` dataset is artificially biased toward the first half of the trajectory, which is over-represented by appearing eleven times more frequently than the second half). Such bias is likely present in real-world CC datasets (in a more subtle way, due e.g. to non-uniform spatial user distribution), where determining bias can be challenging due to the lack of ground truth. With the offline training approach, which trains based on sampling the whole dataset with equal probability, this imbalance leads to overfitting in the first half of the geometric trajectory, while the model fails to learn adequately for the second half. In contrast, this overfitting effect is mitigated in online models thanks to the data-dependent memory updates, which ensure that when the trajectory reaches previously unexplored areas, the model learns these new regions through repeated optimization using fresh samples from this space.

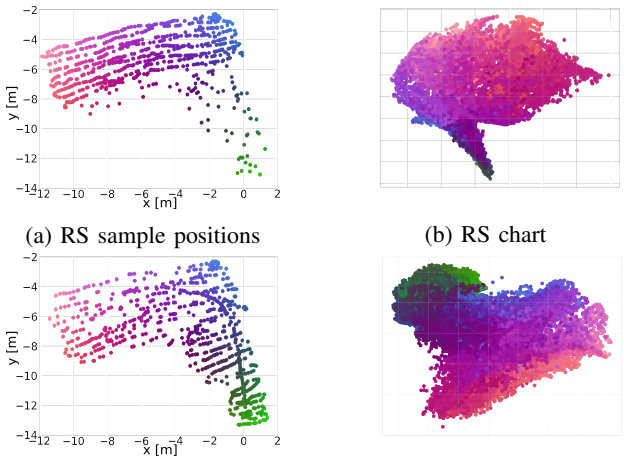
Fig. 2 depicts the ground truth location of samples in final memory and the resulting test CC for the baseline RS method

TABLE I: Model performance based on the replay strategy.

Method	T ↑	C ↑	KS ↓
Offline	73.90 ± 1.12	82.83 ± 1.27	0.61 ± 0.02
RS	87.62 ± 2.39	89.46 ± 1.80	0.47 ± 0.07
Random	83.32 ± 1.10	88.97 ± 1.27	0.41 ± 0.07
SimS-SCN	88.50 ± 0.76	91.85 ± 0.48	$\mathit{0.32 \pm 0.02}$
MMPS	$\mathit{88.85 \pm 1.23}$	$\mathit{92.12 \pm 0.79}$	$\mathbf{0.31 \pm 0.03}$
LDRSS	87.20 ± 1.66	90.97 ± 0.50	0.35 ± 0.02
GDRSS	$\mathbf{90.06 \pm 0.67}$	$\mathbf{92.99 \pm 0.32}$	0.34 ± 0.03

and the best-performing method, GDRSS. The consequence of the equally likely sample selection in RS is evident: the first half of the trajectory (top part in Fig. 2(a)) is oversampled in the geometric space due to being overrepresented in the dataset, while the second half is undersampled. This imbalance has a detrimental impact on the learned CC model, which learns a poor representation of the bottom part of the trajectory (which appears condensed into a small region in the lower left area of the CC in Fig. 2(b)).

In contrast, the final memory obtained using GDRSS demonstrates a more balanced sampling across geometric space (Fig. 2(c)), i.e. it the sample selection process compensates the over-represented part of dataset `cf-02-R`. Consequently, the uneven sampling in the original data (where the first half of the trajectory appears eleven times and the second half only once) does not pose a problem for GDRSS. This is reflected in the resulting CC (Fig. 2(d)), where the second half of the trajectory (towards the green area) is more accurately represented.



(a) RS sample positions (b) RS chart

(c) GDRSS sample positions (d) GDRSS chart

Fig. 2: Last memory sampling positions (left) and color-matched learned CC (right) for RS (top) and GDRSS (bottom).

VI. Conclusion

In this work, we present a novel dual memory framework for online channel charting (CC), introducing various approaches for sample selection within the context of replay-based online continual learning. We evaluated these methods against offline and online baselines, showing that data-dependent sample selection based on gradient-based GDRSS outperforms all other approaches when dealing with non-stationary CSI.

References

[1] C. Studer, S. Medjkouh, E. Gonultaş, T. Goldstein, and O. Tirkkonen, "Channel charting: Locating users within the radio environment using channel state information," *IEEE Access*, vol. 6, 2018.

[2] P. Ferrand, M. Guillaud, C. Studer, and O. Tirkkonen, "Wireless channel charting: Theory, practice, and applications," *IEEE Communications Magazine*, vol. 61, no. 6, pp. 124–130, 2023.

[3] E. Lei, O. Castañeda, O. Tirkkonen, T. Goldstein, and C. Studer, "Siamese neural networks for wireless positioning and channel charting," in *Allerton Conf. on Communication, Control, and Computing*, 2019.

[4] P. Ferrand, A. Decurninge, L. G. Ordoñez, and M. Guillaud, "Triplet-based wireless channel charting: Architecture and experiments," *IEEE Journ. Sel. Areas in Comm.*, vol. 39, no. 8, pp. 2361–2373, 2021.

[5] S. Taner, M. Guillaud, O. Tirkkonen, and C. Studer, "Channel charting for streaming CSI data," in *Asilomar Conference on Signals, Systems, and Computers*, 2023, pp. 1648–1653.

[6] S. Farquhar and Y. Gal, "Towards robust evaluations of continual learning," 2019, arXiv:1805.09733.

[7] D. Feldman, "Core-sets: Updated survey," in *Sampling Techniques for Supervised or Unsupervised Tasks*, F. Ros and S. Guillaume, Eds. Springer International Publishing, 2020, pp. 23–44.

[8] C. Guo, B. Zhao, and Y. Bai, "Deepcore: A comprehensive library for coreset selection in deep learning," in *Database and Expert Systems Applications*. Springer International Publishing, 2022, pp. 181–195.

[9] R. Tiwari, K. Killamsetty, R. Iyer, and P. Shenoy, "GCR: Gradient coreset based replay buffer selection for continual learning," in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 99–108.

[10] Y. Yang, H. Kang, and B. Mirzasoleiman, "Towards sustainable learning: Coresets for data-efficient deep learning," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2023, pp. 39 314–39 330.

[11] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: a strong, simple baseline," in *Proc. Advances in Neural Information Processing Systems*, 2020.

[12] G. Saha, I. Garg, and K. Roy, "Gradient projection memory for continual learning," in *Proc. Int. Conf. on Learning Representations (ICLR)*, 2021.

[13] C. Zhuang, S. Huang, G. Cheng, and J. Ning, "Multi-criteria selection of rehearsal samples for continual learning," *Pattern Recognition*, vol. 132, p. 108907, 2022.

[14] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[15] J. Ju, H. Jung, Y. Oh, and J. Kim, "Extending contrastive learning to unsupervised coreset selection," *IEEE Access*, vol. 10, 2022.

[16] F. Wiewel and B. Yang, "Entropy-based sample selection for online continual learning," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2021, pp. 1477–1481.

[17] F. Euchner, M. Gauger, S. Dörner, and S. ten Brink, "A Distributed Massive MIMO Channel Sounder for "Big CSI Data"-driven Machine Learning," in *25th International Workshop on Smart Antennas*, 2021.

[18] E. Yang, L. Shen, Z. Wang, T. Liu, and G. Guo, "An efficient dataset condensation plugin and its application to continual learning," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[19] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Softw.*, vol. 11, no. 1, p. 37–57, mar 1985.

[20] F. Euchner and M. Gauger, "CSI Dataset dichasus-cf0x: Distributed Antenna Setup in Industrial Environment, Day 1," 2022. [Online]. Available: https://doi.org/10.18419/darus-2854

[21] Y. Vindas and M. Guillaud, "Multi-site wireless channel charting through latent space alignment," in *Proc. International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2024.