IEEE ComSoc IEEE Communications Society | IEEE COMPUTER SOCIETY | IEEE Signal Processing Society | VTS Connecting the Mobile World
IEEE Transactions on Machine Learning in Communications and Networking

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Digital Object Identifier 10.1109/TMLCN.2022.1234567

Multi-Generator Continual Learning for Robust Delay Prediction in 6G

Xiaoyu Lan[1], Jalil Taghia[1], Hannes Larsson[1], Andreas Johnsson[1,2]

[1]Ericsson AB, Ericsson Research, Stockholm, Sweden
[2]Uppsala University, Department of Information Technology, Uppsala, Sweden

Corresponding author: Xiaoyu Lan (email: xiaoyu.lan@ericsson.com).

ABSTRACT In future 6G networks, dependable networks will enable telecommunication services such as remote control of robots or vehicles with strict requirements on end-to-end network performance in terms of delay, delay variation, tail distributions, and throughput. With respect to such networks, it is paramount to be able to determine what performance level the network segment can guarantee at a given point in time. One promising approach is to use predictive models trained using machine learning (ML). Predicting performance metrics such as one-way delay (OWD), in a timely manner, provides valuable insights for the network, user equipments (UEs), and applications to address performance trends, deviations, and violations. Over the course of time, a dynamic network environment results in distributional shifts, which causes catastrophic forgetting and drop of ML model performance. In continual learning (CL), the model aims to achieve a balance between stability and plasticity, enabling new information to be learned while preserving previously learned knowledge. In this paper, we target on the challenges of catastrophic forgetting of OWD prediction model. We propose a novel approach which introducing the concept of multi-generator for the state-of-the-art CL generative replay framework, along with tabular variational autoencoders (TVAE) as generators. The domain knowledge of UE capabilities is incorporated into the learning process for determining generator setup and relevance. The proposed approach is evaluated across a diverse set of scenarios with data that is collected in a realistic 5G testbed, demonstrating its outstanding performance in comparison to baselines.

INDEX TERMS 3GPP, 6G, Continual Learning, Delay Prediction, Generative Replay, Machine Learning.

I. INTRODUCTION

THE evolution towards 6G is expected to enable a new generation of applications, for example in the area of cyber-physical systems [1], [2], demanding ultra-reliable, real-time, and low-latency communication. For such applications, it is crucial to assess performance indicators such as One-Way Delay (OWD) [3] and Round Trip Time (RTT) [4], as even minor disturbances in the communication between two network functions could lead to severe operational and safety risks. Unfortunately, there is complexity associated with assessing the performance outcome of a User Equipment (UE) as it is highly influenced by and dependent on multiple factors such as network configuration, signal conditions, competing traffic, exogenous processes in the UE, and variations in UE hardware [5]. A promising approach, explored in both academia and industry, is based upon Machine Learning (ML) where the performance outcome of a UE, such as the OWD [6]–[8], is predicted based upon statistics available to the network operator. Accurate performance prediction, not only with respect to point estimates but also with respect to distributions and tails [7], [9], [10], is essential for proactive service assurance, troubleshooting, verification, and performance optimization.

Previous research has demonstrated the feasibility of modeling delay; however, it has also shown that model performance deteriorates over time due to distributional shifts in data. These shifts are due to factors such as previously unseen user behavior patterns (e.g., traffic fluctuations, mobility) and variations in UE chip sets [11]. Various mitigation strategies have been explored, including transfer learning and domain adaptation [12], and semi-supervised learning [13]. More broadly, it is well established that network management models degrade over time due to the dynamic nature of networks and their underlying compute resources [14]. The challenge of maintaining network management models over time has also been identified in standardization, and thus there is an increased interest in incorporating automated ML workflows and continuous operations into the network

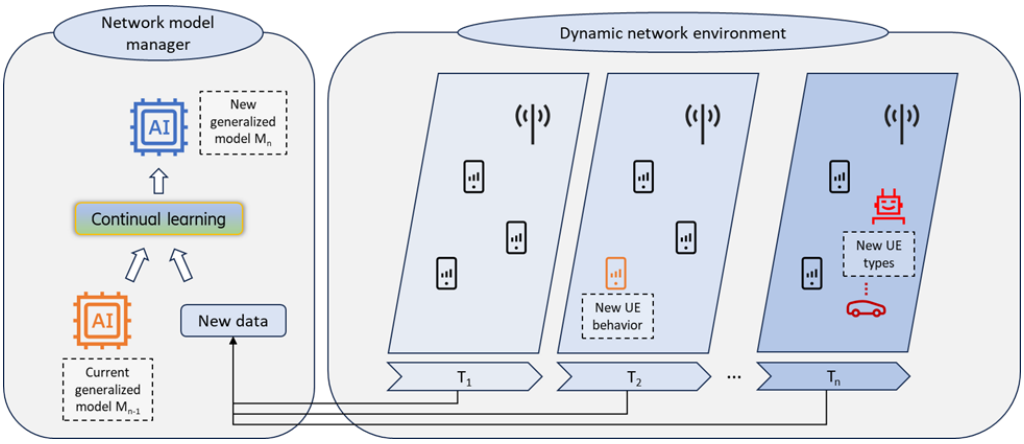

FIGURE 1. A dynamic network environment gives rise to distributional shifts in the data, causing loss of OWD model performance. Continual learning can balance model plasticity and stability, ensuring new information is captured in the model while preserving old knowledge, that is avoiding catastrophic forgetting.

architecture. For example, in 3GPP, the ML workflow for the Network Data Analytics Function (NWDAF) is specified in [15], while the O-RAN ML workflows are reviewed in [16].

Despite recent advancements in model generalization, a significant challenge remains: sustaining model performance over extended periods. This challenge is illustrated in Fig. 1, where a network environment evolves over time, introducing distributional shifts that can degrade model performance. Each time step in this dynamic environment can be considered as a continual learning (CL) task, new data is observed and thereafter integrated with the existing generalized model through CL, producing an updated generalized model. The key challenge lies in balancing model plasticity with learning stability [17] - ensuring that new behaviors in the network environment are learned while preserving previously acquired knowledge. In other words, preventing catastrophic forgetting in ML [18]. This paper specifically addresses this challenge for OWD prediction models, as discussed above, and proposes a novel approach based on CL. The method leverages generative replay [19] and extends such approaches with a multi-generator framework, where generators are designed and maintained using domain knowledge of the network and its UEs. Generative replay, specifically the proposed multi-generator approach, not only mitigates catastrophic forgetting, it also improves performance on the distribution tail as well as enabling network operators to reduce the storage requirements for previously seen data samples.

The main contributions of this paper are as follows. We propose a novel method for CL based on generative replay, introducing the concept of multi-generator in combination with tabular variational autoencoders (TVAE) to mitigate catastrophic forgetting in OWD prediction. Additionally, we elaborate on a method for determining generator setup and relevance, leveraging knowledge of UE capabilities, thereby incorporating domain knowledge into the learning process. The proposed approach is evaluated across a wide range of scenarios in a realistic 5G testbed, demonstrating its effectiveness in comparison to a baseline both from a point estimate perspective, and also on the tail. Moreover, we highlight the reduced need for data storage, targeting the challenges of resource constraints in 5G networks, which enhances the efficiency of the method. Finally, we discuss how an OWD prediction engine can be integrated into the 3GPP architecture, specifically within NWDAF, thereby simplifying the realization of the approach in operational networks.

The rest of the paper is organized as follows. Section II describes the background on our use case and problem formulation. Section III presents the necessary background on deep generative replay and introduces the novel approach on multi-generator CL, whereas Section IV describes our testbed and datasets. Section V describes an in-depth evaluation framework of the approach and Section VI presents the evaluation results. In Section VII, we provide discussions related to the approach and an architectural view on OWD prediction in 3GPP. Section VIII contains related work, and conclusions are found in Section IX.

II. PROBLEM DEFINITION

In this section, we provide a high-level background on OWD prediction, proceed with notations and framework of CL, and present problem statement.

A. BACKGROUND

The objective of this paper is to establish a robust approach for training and maintaining a ML model that predicts OWD as experienced by a UE. This prediction is based on measurements taken with high frequency from the baseband in the Radio Access Network (RAN). Although our previous work [11] addressed some aspects of OWD model generalization, the expanded aim of this paper is to prevent catastrophic forgetting, and thus ensure robustness during long-term maintenance of OWD models.

In the scenario illustrated in Fig. 1, a model manager is tasked with maintaining a OWD model, which was originally trained using historical data. These data distributions are influenced by a variety of factors, such as radio conditions, network configurations, network load, UE movement patterns, and UE device types. As the network evolves over time, the resulting dynamicity leads to distributional shifts in the measurements; among others, the dynamicity is related to new UE behavior, previously unseen UE types, and potentially altered network configurations. Consequently, in the subsequent time slot, the OWD and baseband features exhibit new distributions.

The challenge targeted in this paper is to maintain the ML model performance over time, leveraging and extending the concept of CL for retraining the model with new data while preserving knowledge related to previous steps.

B. NOTATIONS AND FRAMEWORK

In the following, we describe the notations and framework. Here, the task is prediction of OWD as observed by the UE using RAN data collected in the baseband.

We denote the ML task by T. The OWD measurements from the UE are denoted by y distributed according to $P(Y)$. Measurements from the RAN are referred to as the features. Let x_j denote the j-th feature distributed according to $P(X_j)$. A collection of d different features are shown for convenience in the vector notation by $\boldsymbol{x} \in \mathbb{R}^d$; however, note that no explicit spatial information is assumed across the elements of $\boldsymbol{x}$, indicating that we are concerned with tabular data.

We assume a CL scenario where the underlying ML task, T, remains the same. However, the task condition may change due to the effect of external factors that result in distributional shift in space of X and Y. In the context of OWD prediction, example of such external factors are the UE's device type along with the UE position and movement patterns (refer to Table 1). Let $T^{(i)}$ denote the task at the i-th condition. A set of such task conditions are denoted by $\mathcal{T} = \{T^{(i)} \mid i = 1, 2, \ldots, I\}$, where I is the number of tasks. In CL framework, the set $\mathcal{T}$ is treated as a sequence.

Let $\mathcal{D}_{T^{(i)}} = \{\boldsymbol{x}_n^{(i)}, y_n^{(i)} \mid n = 1, \ldots, N^{(i)}\}$ denote our dataset consisting of pairs of input features and output targets at the i-th task condition $T^{(i)}$, where N is the number of samples in the dataset. Further, let M denote a ML model that serves as the predictive model of y given $\boldsymbol{x}$. In the CL framework, task conditions are introduced sequentially which points at the sequential learning of the predictive model.

Let θ denote the parameter set of the predictive model. The learning begins with the dataset from a given task condition, $M\left(\mathcal{D}_{T^{(i)}}; \theta\right)$ parameterized by θ, and follows with the next task condition, $M\left(\mathcal{D}_{T^{(i+1)}}; \theta\right)$. However, a major complication is that the datasets from two arbitrary task conditions, $\mathcal{D}_{T^{(i)}}$ and $\mathcal{D}_{T^{(i+1)}}$, may not share the same underlying distributions, that is $P(X_j^{(i)}) \neq P(X_j^{(i+1)})$ for the j-th feature and $P(y^{(i)}) \neq P(y^{(i+1)})$. As the result, the learned model would have a reduced predictive relevance for data from the i-th task condition. The problem is commonly referred to as the *catastrophic forgetting* which magnifies as sequential learning continues.

One line of approaches for solving the catastrophic forgetting problem is to refrain the model from forgetting by presenting data, or a subset of data, from the past task conditions, referred to as the *data replay*. While highly effective, the assumption of having access to the past data, or a subset of which, is restrictive for many practical applications, especially in resource-constrained telecom network environments, due to the cost of data storage as well as privacy concerns.

C. PROBLEM STATEMENT

The *deep generative replay* framework [19] presents a promising approach to CL, addressing the limitations associated with the data replay. While potentially effective, *we argue that the generative replay itself can suffer from catastrophic forgetting*. As the number of tasks increases over time, with the growing diversity resulting from introduction of new tasks, the generative model tends to capture merely the modalities in the data that represent the bulk of the underlying data distribution and fail to capture all the modalities. We refer to this phenomenon as *mode collapse* that is when a modality that was present once collapses into a more dominant modality.

The main reason for the mode collapse is the limited expressiveness capability of the underlying generative model. One way to improve the model expressiveness is to choose a generative model suitable for the data under consideration. We show that this has a major influence in the overall success of the generative replay in the framework of CL. Specifically, in the context of OWD prediction that is concerned with tabular data, we show that using a generative model tailored for tabular data substantially improves the performance. However, this does not resolve the mode collapse if the data generation process differs considerably across tasks. As an example, the UE device type is a factor that gives rise to different data generation processes. In this regard, we argue that having multiple generative models where each represents one of the modalities of their respective data generation processes can further improve the overall model expressiveness. However, the question is how one would select the relevant generative model from a given collection of generative models? This points to a need for a generator selector. One approach is to construct the generator selector in a data-driven fashion. Alternatively, one can leverage domain knowledge in construction of the generator selector. As an example, in the context of our running example, understanding of the UE device type is knowledge that can help select the right generator. Here we show that incorporating domain knowledge can significantly mitigate catastrophic forgetting by providing contextual frameworks

that guide learning processes. By embedding these domain-specific insights into the learning architecture, models can develop more stable representations that reduce the risk of mode collapse (or in other words catastrophic forgetting) as a new task arises.

In summary, in the context of OWD prediction, we make the following contributions to reduce risks of catastrophic forgetting: (1) use of a generative model that is tailored for tabular data, (2) construction of a multi-generator architecture for the generative replay, and (3) devising a domain-guided generator selector. We show via empirical experiments that the proposed steps substantially reduce risks of catastrophic forgetting and contribute to improving model's ability to maintain performance across diverse tasks.

III. MULTI-GENERATOR CONTINUAL LEARNING

In this section, we describe our novel domain-guided approach to construction of a multi-generator-based generative replay in the framework of CL. We start this section with a background of the preliminaries needed for our construction and proceed with the proposed approach.

A. BACKGROUND

1) Deep generative replay

The model architecture of the deep generative replay consists of a generative model, *generator*, and a task solving model, *solver*. Shin et al. in [19] refer to this dual model architecture as *scholar*, defined by $\mathcal{H} = \{G, S\}$ where G denotes the generator and S denotes the solver. The generator is a parameterized generative model, with a parameter set denoted by ϕ, that is learned to produce real-like samples of the input features, and the solver is a parameterized predictive model of the task with a parameter set denoted by θ.

The components of the scholar, generator and solver, are learned in a two-step sequential-learning framework. Let $\mathcal{H}^{(i)}$ denote the scholar model at the i-th task condition. The generator and the solver of $\mathcal{H}^{(i)}$ are denoted by $G(\phi^{(i)})$ and $S(\theta^{(i)})$. Furthermore, for convenience, we use the following notations equivalently: $G^{(i)} \equiv G(\phi^{(i)})$ and similarly, $S^{(i)} \equiv S(\theta^{(i)})$.

The training procedure involved in learning of the current scholar, $\mathcal{H}^{(i)}$, from the prior scholar, $\mathcal{H}^{(i-1)}$, involves two *independent* training procedures for the generator and the solver.

In the first step, from the prior state of the generator, $G^{(i-1)}$, a set of artificial input data samples are produced, referred to as the replay input features and denoted by $\boldsymbol{x}'$. The replay targets, y', are predicted from the prior state of the solver, $S^{(i-1)}$, given replay inputs, $\boldsymbol{x}'$, according to: $y' = S\left(\boldsymbol{x}'; \theta^{(i-1)}\right)$, where $\theta^{(i-1)}$ denotes the prior state of the solver's parameter set. The replay targets are the solver's response to the replay input in the past. Finally, given the input features from the current task condition, $\boldsymbol{x}$, the targets, y, and the replay targets, y', a training loss is constructed according to:

$$\ell_S = \alpha \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_{T^{(i)}}} \left[L_S \left(S\left(\boldsymbol{x}; \theta^{(i)}\right), y \right) \right] + (1-\alpha) \mathbb{E}_{\boldsymbol{x}'\sim G^{(i-1)}} \left[L_S \left(S\left(\boldsymbol{x}'; \theta^{(i)}\right), y' \right) \right], \quad (1)$$

where $\boldsymbol{x}' \sim G^{(i-1)}$ denotes the replay inputs drawn from the prior generator, $(\boldsymbol{x}, y) \sim \mathcal{D}_{T^{(i)}}$ denotes the pair of input samples and targets taken randomly from the dataset of the i-th task condition, α is a ratio of mixing real data from the current task condition with real-like data from the previous task condition and L_S denotes the solver's loss function. This step updates the solver of the i-th scholar.

Independent from the first step, in the second step, the replay inputs $\boldsymbol{x}'$ are mixed with the input features from the current task condition, $\boldsymbol{x}$. Let $\widetilde{\boldsymbol{x}} := (\boldsymbol{x}, \boldsymbol{x}')$ define the concatenation of the replay inputs with the current input features. Given $\widetilde{\boldsymbol{x}}$, the generator is trained (or adapted) to learn to generate samples from their cumulative underlying distribution. The learning step involves minimization of the generator's objective loss function,

$$\ell_G(\phi) = \mathbb{E}_{\boldsymbol{x}'\sim G^{(i-1)}, \boldsymbol{x}\sim\mathcal{D}_{T^{(i)}}} \left[L_G \left(\hat{\widetilde{\boldsymbol{x}}}, \widetilde{\boldsymbol{x}}; \phi^{(i)} \right) \right], \quad (2)$$

where $\boldsymbol{x}' \sim G^{(i-1)}$ denotes the replay inputs drawn from the prior generator, $\boldsymbol{x} \sim \mathcal{D}_{T^{(i)}}$ denotes the input samples taken randomly from the dataset of the i-th task condition, L_G is the loss function of the generator, and $\hat{\widetilde{\boldsymbol{x}}}$ is the reconstructed samples produced by the generator given the current setting of its parameter set, $\phi^{(i)}$. This step updates the generator of the i-th scholar.

2) Tabular Variational Auto-Encoders

TVAE is the state-of-the-art variational auto-encoder (VAE) for tabular data generation [20]. It is specifically designed for tabular data, which is typically composed of a mix of both discrete and continuous features. Continuous features may have multi-modal and non-Gaussian distributions whereas discrete features are sometimes imbalanced making the modeling difficult.

The basic scheme of a VAE composes of an encoder and a decoder [21]. The encoder compresses the input $\boldsymbol{x}$ into the latent space. The decoder receives as input the information sampled from the latent space and produces $\boldsymbol{x}'$ as similar as possible to $\boldsymbol{x}$. In TVAE, the modeling for encoder is similar to conventional VAE. The decoder is designed specially so that the probability distribution of data can be modeled accurately; refer to [20] for details. Also, a mode-specific normalization is designed to deal with features with complicated distributions. This architecture allows TVAE to generate data that closely match the distribution of real tabular data, making it more suitable for applications like data imputation, synthetic data generation, and tabular data modeling.

B. PROPOSED SOLUTION: MULTI-GENERATOR GENERATIVE REPLAY

This section describes our construction of multi-generator generative replay. We first describe the construction of the generator selector and proceed with describing the construction of the multi-generator generative replay.

1) Generator relevance determination

Let $\boldsymbol{a} = (a_1, \ldots, a_J)^\top$, for all $a_j \in \{0, 1\}$, denote a binary vector of J data configurations at a given task condition where each element specifies a different configuration, referred to as the *task configuration vector*. In [11], Rao et al. showed that OWD distribution is significantly dependent upon the UE type. Therefore, in our use case of OWD prediction, the elements of the task configuration vector represent UE device type; as an example, for three different UE types as described in Table 1, the task configuration vector is expressed by a 3-dimensional vector with binary elements with the three elements corresponding to the UE types,

$$\boldsymbol{a} = (\text{UE1}, \text{UE2}, \text{UE3})^\top. \tag{3}$$

Further, we denote the task configuration vector at the i-th task condition by $\boldsymbol{a}^{(i)}$.

In a similar fashion, we define a configuration vector for the generators, named the *generator configuration vector*. Formally, let $\boldsymbol{b}_k = (b_{k,1}, \ldots, b_{k,J})^\top$, for all $b_{k,j} \in [0, 1]$, denote the configuration vector for the k-th generator with the elements representing the same configurations as in the task configuration vector. However, unlike $\boldsymbol{a}$, elements of $\boldsymbol{b}$ do not need to be necessarily binary and can take soft values between zero and one; if the generator is fully applicable for a given configuration its corresponding element in $\boldsymbol{b}$ is set to one, if it is not applicable it is set to zero, and otherwise it may be set to a value in-between zero and one.

We then define a vector of relevance scores for the i-th task condition, referred to as the *relevance vector* and denoted by $\boldsymbol{r}^{(i)} \in \mathbb{R}^K$ where K is equal to the number of generators. The elements of the relevance vector quantify how well a generator is relevant to the given task condition, and it is computed by the inner product of the task configuration vector and the generator configuration vector, expressed as:

$$\boldsymbol{r}^{(i)} = \left(\boldsymbol{b}_1{}^\top \boldsymbol{a}^{(i)}, \ldots, \boldsymbol{b}_K{}^\top \boldsymbol{a}^{(i)}\right)^\top. \tag{4}$$

Finally, the index of the most relevant generator is given by:

$$k_* = \underset{k}{\operatorname{argmax}} \left(\boldsymbol{r}^{(i)}\right), \quad \forall k = 1, \ldots, K. \tag{5}$$

In the context of our use case, from domain knowledge, we hypothesize that the UE device type is the major differentiating factor of data generation processes. As such for the first UE, the generator configuration vector is set to $\boldsymbol{b}_1 = (1, 0, 0)$ which implies that this generator is suitable for the first UE device type. Similarly for the second and third UE devices, the generator configuration vectors are defined as $\boldsymbol{b}_2 = (0, 1, 0)$ and $\boldsymbol{b}_3 = (0, 0, 1)$, respectively. As an example, let the task configuration vector at the current task be denoted by $\boldsymbol{a} = (0, 1, 0)$ indicating the second UE device type. Then following (4) and (5), the generator defined by $\boldsymbol{b}_2$ would be the most relevant to the task condition defined by $\boldsymbol{a}$. In a similar fashion, upon arrival of a new task, the relevant generator will be selected based on the setting of its task configuration vector.

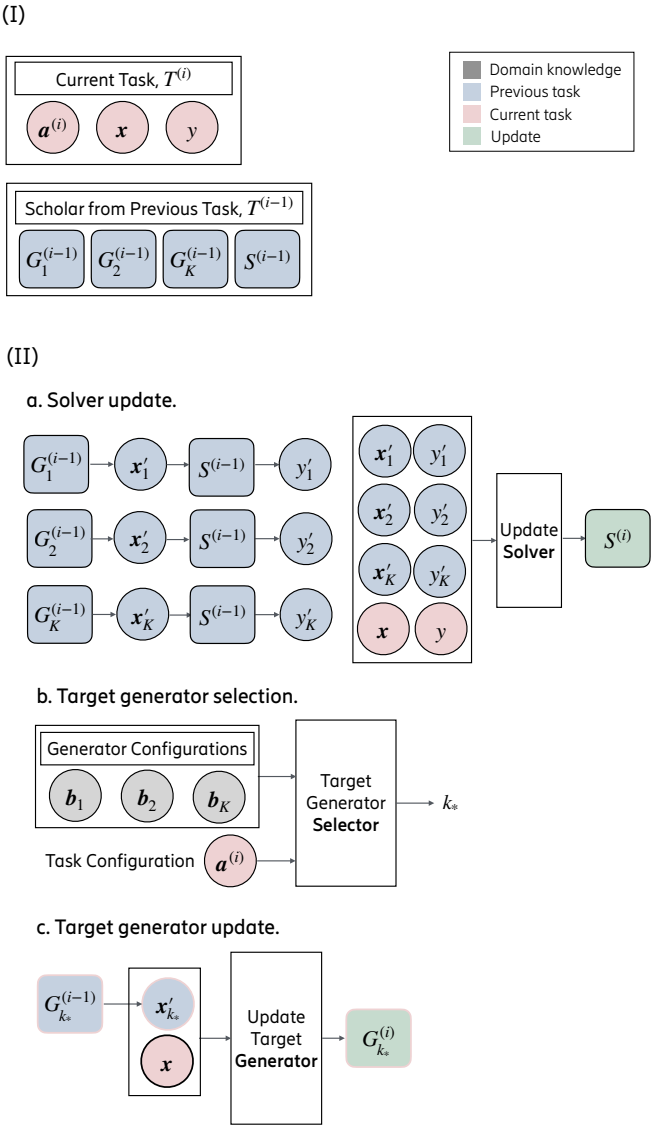

FIGURE 2. Multi-generator generative replay architecture. (I) Scholar from the previous task, and inputs and targets in the current task. (II) The procedure of learning of the scholar at the current task, which involves three steps.

2) Construction of multi-generator generative replay

We revisit the construction of the generative replay and extend it from a single generator to a finite set of multi-generators.

We define a scholar by a set of generators and a single solver, $\mathcal{H} = \{\{G_k\}_{1:K}, S\}$. Further, we assume their corresponding set of generator configuration vectors, $\{\boldsymbol{b}_k\}_{1:K}$, are given. The learning of the scholar at the current task condition, $\mathcal{H}^{(i)}$, involves a procedure of updating the solver followed by a generator selection step and a procedure of updating the generator. Fig. 2 summarizes the proposed architecture for multi-generator generative replay with three steps procedure of learning of the scholar at the current task which is described in the following.

a: Solver update

Let $\{\boldsymbol{x}'_k\}_{1:K}$ denote a set of replay inputs generated from $\{G_k^{(i-1)}\}_{1:K}$. A set of corresponding replay targets, $\{\boldsymbol{y}'_k\}_{1:K}$, are predicted from the prior state of the solver, $S^{(i-1)}$, where $y'_k = S\left(\boldsymbol{x}'_k; \theta^{(i-1)}\right)$. Finally, given the input features from the current task condition, $\boldsymbol{x}$, the targets, y, and the replay targets, the solver's training loss is constructed according to:

$$\ell_S = \alpha \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_{T^{(i)}}}\left[L_S\left(S\left(\boldsymbol{x};\theta^{(i)}\right), y\right)\right] + (1-\alpha)\mathbb{E}_{\left\{\boldsymbol{x}'_k\sim G_k^{(i-1)}\right\}_{1:K}}\left[L_S\left(S\left(\boldsymbol{x}'_k;\theta^{(i)}\right), y'_k\right)\right]. \quad (6)$$

b: Target generator selection

We need to obtain the index of the most relevant generator, k_*, for the current task configuration vector, $\boldsymbol{a}^{(i)}$, given $\{\boldsymbol{b}_k\}_{1:K}$. This is done by application of Eq. (4) and Eq. (5). The corresponding generator is referred to as the target generator at the i-th task condition and it is denoted by $G_{k_*}^{(i-1)}$.

c: Target generator update

First, from the prior state of the target generator, $G_{k_*}^{(i-1)}$, replay input features are produced which are denoted by $\boldsymbol{x}'_{k_*}$, where the subscript k_* is used to emphasize on the generation through the target generator. The replay inputs are mixed with the input features from the current task condition, $\boldsymbol{x}$, and form $\widetilde{\boldsymbol{x}} := (\boldsymbol{x}, \boldsymbol{x}'_{k_*})$. Given $\widetilde{\boldsymbol{x}}$, the target generator is updated via minimization of the following objective function:

$$\ell_{G_{k_*}}(\phi_{k_*}) = \mathbb{E}_{\boldsymbol{x}'_{k_*}\sim G_{k_*}^{(i-1)}, \boldsymbol{x}\sim\mathcal{D}_{T^{(i)}}}\left[L_{G_{k_*}}\left(\hat{\widetilde{\boldsymbol{x}}}, \widetilde{\boldsymbol{x}}; \phi_{k_*}^{(i)}\right)\right], \quad (7)$$

where $\hat{\widetilde{\boldsymbol{x}}}$ is the reconstructed samples produced by the target generator given the current setting of its parameter set, $\phi_{k_*}^{(i)}$. This step updates the target generator of the i-th scholar while the other generators are unaltered.

IV. A 5G TESTBED FOR CREATION OF ONE-WAY DELAY DATASETS

To evaluate the multi-generator generative replay approach for robust OWD prediction in future 6G networks, we leverage an in-house 5G-mmWave testbed [11] to create datasets consisting of base station features and ground-truth OWD values. In the following, while we briefly describe the testbed, we focus on the experiments conducted in support of evaluating our approach.

The testbed is a 5G NSA system build upon commercially available Ericsson StreetMacro 6701 device that implements 5G NR, and an Ericsson Radio Dot 2243 device that implements 4G LTE. The StreetMacro does analog beamforming using one horizontal and one vertical beam (1 × 200 MHz dual polarized beams). We set the 4G LTE eNB to operate on band B3 (1800 MHz frequency, with 5 MHz bandwidth), and 5G NR gNodeB (gNB) on band n257 (28 GHz frequency, with 100 MHz bandwidth). The testbed resides in an indoor testbed area. The floor plan and associated positions of the 5G NR and 4G LTE radios, is illustrated in Fig. 3. For details regarding the setup we refer to [11].

Similarly to [11], we measure OWD, between the UE and a receiver in the network, with three different types of 5G UE devices, with different manufacturers and chipsets, referred to as UE1, UE2, UE3. The OWD measurements are conducted using TWAMP [22] probe packets of sizes 100 and 1400 bytes. An additional UE is configured for creation of network traffic scenarios, where that UE is introducing competing uplink traffic into the network, ranging from 0 to 60 Mbps, in steps of 10 Mbps. A UE is either stationary (position 1, 2, or 3), or moving according to a set of predefined movement patterns (rectangular, zigzag, and linear). The experiment parameters are summarized in Table 1.

The experimental scenarios were designed to expose the UE and the network to a wide range of conditions that contribute to variations in OWD and baseband features. The range of loads was selected to expose the network from being completely empty, to operating in a state of congestion. In addition, the UE movement patterns were designed to ensure that we captured excellent channel conditions, worst-case cell edge conditions, as well as conditions in between. These network conditions, when put into a sequence, represent distributional shifts that the multi-generator generative replay approach must mitigate.

In addition to collection of ground-truth OWD, the testbed was configured with multiple measurement points to extract network metrics [11]. More specifically, we extract logs from the gNB corresponding to beam forming, UE connectivity, and uplink (UL)/downlink (DL) scheduling events [23]–[25], on the sub-millisecond scale. Of a much larger set available, we choose to monitor and use 103 metrics.

For each set of parameters listed in Table 1, we sent 5000 TWAMP packets, spacing each packet 50 ms apart. This process generated a dataset with 1.26 million accurate OWD samples. We averaged the network metrics obtained from the gNB over 50 ms, and organized them into bins aligned in time with the OWD values. Consequently, each sample in the dataset includes a OWD measurement and the average value of each feature within the 50 ms bin.

In this paper, we focus on OWD prediction for UL, while the approaches can be generalized to DL as well. Relevant network metrics are used as features. We use measurement

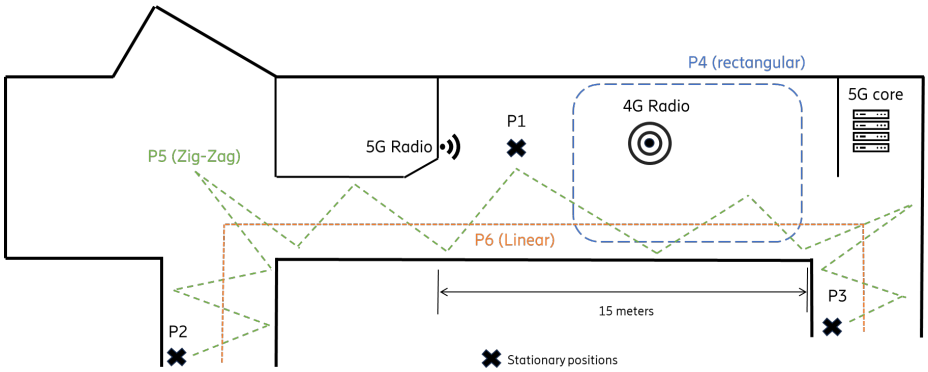

FIGURE 3. **Floor plan of the testbed area illustrating positions and movement patterns (P1 - P6).**

TABLE 1. **Testbed experiment parameters.**

UE device type	UE 1, UE 2, UE 3
Probe pkt. size	100 B, 1400 B
UL load	0, 10, 20, 30, 40, 50, 60 Mbps
Position and Movement patterns	P1: Stationary at positions 1
	P2: Stationary at positions 2
	P3: Stationary at positions 3
	P4: Rectangular movement
	P5: Zigzag movement, pos. $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$
	P6: Linear movement, pos. $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$

experiments data with all varying UL load and large probe packet size (1400B).

V. EVALUATION METHODOLOGY

In this section, we describe our evaluation methodology, starting with describing our approach in designing CL task sequences, presenting methods considered in our evaluation framework, and introducing relevant metrics devised for assessing the performance of the CL methods.

A. DESIGNING CL TASK SEQUENCES

In [11], Rao et al. showed that various configuration settings such as UE device type and the UE movement pattern can result in distributional shift both at the input space as well as in the task space (OWD). Inspired by this, we create a series of CL tasks by utilizing configuration settings from various experiments to define the tasks. In particular, each task is constructed based on the UE type, along with its position and movement pattern as defined and illustrated in Table 1 and Fig. 3. Accordingly, two groups of CL task sequences are constructed which are shown in Table 2.

In the first group, named Group 1, we assume there are only two different UE device types along with 6 different positions and movement patterns. Table 2 (a) summarizes 6 different scenarios considered in our evaluations each labeled by a case identifier (Case ID). The scenarios are devised based on the UE device types and their movement patterns. As an example, in one scenario (Case ID 1), UE 1 and UE 2 are considered together with the UE movement pattern from stationary to moving ($S \rightarrow M$). To explore the impact of task order on the evaluation results, in the other scenario (Case ID 2), the order of task sequence of UE 1 and UE 2 is reversed from moving to stationary ($M \rightarrow S$). In the same way, the task sequence of UE 2 and UE 3 (Case IDs 3,4) and UE 3 and UE 1 (Case IDs 5,6) are designed. All scenarios in Group 1 consists of 12 CL tasks in each. Among the various order of tasks we explored, the ones presented here are representative of the overall patterns observed.

Similarly, in the second group, named Group 2, we assume there are three different UE device types with 6 different positions and movement patterns. Two scenarios (Case IDs 7,8) that are considered in the evaluations are summarized in Table 2 (b). Both scenarios in Group 2 consists of 18 CL tasks in each.

B. EVALUATION FRAMEWORK

We evaluate our multi-generator generative replay approach using the two defined groups of CL task sequences, and compare to several baselines as described below. For approaches based on generative replay, we employ a fully connected MLP neural network as the solver. The architecture of the MLP is shown in Table 3. In the following, we discuss 4 baseline approaches, as well as providing additional details regarding our own approach.

1) Baseline approaches

a: Naïve

As in [27], Naïve method could be used as the lower bound baseline for stability evaluation, in which the model is trained continuously with data of each task, without any particular framework to control forgetting. That is, the solver is finetuned using data from the new task, without access to the data from the previous tasks.

b: Cumulative

Cumulative method is used to evaluate the upper bound of forgetting in CL as in [27]. For each task, it accumulates all data from previous tasks and data of current task, re-train the solver from scratch. There is no forgetting as cumulative method could access and use all the previous data. However, it is often infeasible in real world applications, due to constrains in data storage and privacy.

TABLE 2. CL task sequences for Group 1 and Group 2. Each task sequence is labeled by Case ID and Case Name. CL tasks are devised based on the UE device types, along with its position and movement pattern. Note, P is for position and movement pattern, whereas S is for stationary, and M for moving.

(a) Group 1

Case ID	Case Name	CL task	1	2	3	4	5	6	7	8	9	10	11	12
1	UE 1,2	UE	2	1	2	1	2	1	2	1	2	1	2	1
	$S \rightarrow M$	P	1	1	3	3	2	2	4	4	6	6	5	5
2	UE 1,2	UE	1	2	1	2	1	2	1	2	1	2	1	2
	$M \rightarrow S$	P	5	5	6	6	4	4	2	2	3	3	1	1
3	UE 2,3	UE	2	3	2	3	2	3	2	3	2	3	2	3
	$S \rightarrow M$	P	1	1	3	3	2	2	4	4	6	6	5	5
4	UE 2,3	UE	3	2	3	2	3	2	3	2	3	2	3	2
	$M \rightarrow S$	P	5	5	6	6	4	4	2	2	3	3	1	1
5	UE 1,3	UE	3	1	3	1	3	1	3	1	3	1	3	1
	$S \rightarrow M$	P	1	1	3	3	2	2	4	4	6	6	5	5
6	UE 1,3	UE	1	3	1	3	1	3	1	3	1	3	1	3
	$M \rightarrow S$	P	5	5	6	6	4	4	2	2	3	3	1	1

(b) Group 2

Case ID	Case Name	CL task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
7	UE 1,2,3	UE	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1
	$S \rightarrow M$	P	1	1	1	3	3	3	2	2	2	4	4	4	6	6	6	5	5	5
8	UE 1,2,3	UE	1	3	2	1	3	2	1	3	2	1	3	2	1	3	2	1	3	2
	$M \rightarrow S$	P	5	5	5	6	6	6	4	4	4	2	2	2	3	3	3	1	1	1

TABLE 3. Neural network architecture used in evaluations.

Generator VAE	Encoder: Hidden layers sizes: (128, 128). Activation function: ReLU. Decoder: Hidden layers sizes: (128, 128). Activation function: ReLU.
Generator TVAE	The TVAE Synthesizer in python library SDV [26] is used to train a TVAE model and generate synthetic data. Default parameters are applied, e.g. size of each hidden layer in the encoder and decoder is (128, 128).
Solver MLP	Hidden layers sizes: (200, 150, 100, 50). Activation function: ReLU. Loss function: MSE loss. Optimizer: Adam. Learning rate: 0.001.

c: Single generator with VAE (SingleGen-VAE)

It is illustrated in the state-of-the-art deep generative replay algorithm [19] that a VAE could be used as the generator. We evaluate its performance on our cases. The architecture of the VAE is shown in Table 3. It is worth noting that we experimented with different numbers of layers and layer sizes where all of which produced similar results.

d: Single generator with TVAE (SingleGen-TVAE)

As it is presented in Section III, TVAE is specifically designed for tabular data generation. The dataset of this study is created from the 5G testbed, and the features of relevant network metrics are a mix of both discrete and continuous columns. Considering the tabular nature of the data, we view TVAE as a more appropriate choice compared to VAE. This aligns with one of the objectives of this work, which is to highlight the significance of selecting the right generative model based on the data structure. The architecture of the TVAE is shown in Table 3.

2) Multi-generator with TVAE (MultiGen-TVAE)

As it is illustrated in Section III, based on domain knowledge, UE device type is the primary determinant of how data is generated. The generator configuration vectors $\boldsymbol{b}_1 = (1, 0, 0)$,$\boldsymbol{b}_2 = (0, 1, 0)$ and $\boldsymbol{b}_3 = (0, 0, 1)$ are defined for three UE device types as described in Section III. For CL task sequence Group 1, there are only two different UE device types in each scenario, which leads to two generators, one for odd-numbered tasks and the other for even-numbered tasks. For CL task sequence Group 2, there are three different UE device types in each scenario, which leads to three generators. Here we consider TVAE as our choice of the generative model.

C. PERFORMANCE EVALUATION

As the prediction of OWD is a regression task, the performance of the model is evaluated using a standard percentage error metric known as mean absolute percentage error (MAPE), aligning with our prior study [11], thereby facilitating consistent benchmarking and comparative analysis. MAPE is defined as:

$$\text{MAPE} = \frac{100}{N} \sum_{i=0}^{N-1} \frac{|y_i - \widehat{y}_i|}{|y_i|}, \tag{8}$$

where y_i is the ground truth, $\hat{y}_i$ is the predicted OWD, and N is the number of samples in the task dataset. Lower MAPE values indicate better model performance.

To evaluate the performance of the selected CL algorithms, we split the dataset for each CL task into training and test sets (70% for training). Training is done sequentially with the training set of each task according to the task sequence. Performance is then evaluated using the test sets for all I tasks. After the model finishes learning of all I tasks, we get the result matrix $R \in \mathbb{R}^{I,I}$, where $R_{i,j}$ is the performance of the model on test set of task T_j after training with the training set of task T_i. Specifically for our evaluations, the model performance $R_{i,j}$ is MAPE values. Inspired by the two CL evaluation metrics Average Accuracy (ACC) and Backward Transfer (BWT) in [28], we define metrics Average MAPE (AveMAPE) and Forgetting(F) for our regression task OWD prediction, where

$$\text{AveMAPE} = \frac{1}{I}\sum_{i=1}^{I} R_{I,i}, \tag{9}$$

$$\text{F} = \frac{1}{I-1}\sum_{i=1}^{I-1} R_{I,i} - R_{i,i}. \tag{10}$$

AveMAPE evaluates the overall performance of the tasks learned so far. F shows the influence of learning the current task on the performance of the previous tasks. In other words, it evaluates the memory stability of old tasks. Higher values of F indicate increased forgetting.

Furthermore, here, we introduce a new metric to assess effect of both short-term and long-term forgetting, since F alone does not capture short-term and long-term forgetting specifically. We refer to this metric as F_k, which effectively measures average k-step forgetting and is defined as:

$$\text{F}_k = \frac{1}{I-k}\sum_{i=1}^{I-k} R_{i+k,i} - R_{i,i}, \forall k = 1, \ldots, I-1, \tag{11}$$

where for smaller values of k, F_k reflects short-term forgetting while for larger values of k, it captures the long-term forgetting.

VI. RESULTS

In this section, we first present the evaluation results of the CL task sequences Group 1 and Group 2, defined in Section V. To further analyze our observations, in the following, we evaluate the distribution of generated $\boldsymbol{x}'$ and corresponding y' by VAE and TVAE, single generator and multi-generator. In addition, we evaluate the model performance on the tail of the OWD distribution. Finally, the model performance of single generator and multi-generator with the equal number of learnable parameters are compared.

A. EVALUATION ACROSS GROUP 1 AND GROUP 2

Fig. 4 summarizes the model performance across methods, our multi-generator approach as well as the baselines. The performance of the model is evaluated by the metrics AveMAPE, F, and selected F_k reflecting of short-term and long-term forgetting: namely, F_4 and F_8 for Group 1 with 12 tasks in each sequence, and F_6 and F_{12} for Group 2 with 18 tasks in each sequence.

The results notably demonstrate that our proposed multi-generator based approach, MultiGen-TVAE, consistently outperforms the single-generator based approaches, SingleGen-TVAE and SingleGen-VAE, across all metrics and scenarios. Next, we observe that the choice of generative model considerably impacts overall performance; notably, the TVAE-based approach SingleGen-TVAE clearly outperforms the alternative VAE-based approach SingleGen-VAE. This suggests that VAE struggles to generate sufficiently representative data samples for previous tasks, potentially introducing noise with the generated samples. Additionally, as expected, the figures show that the Naïve method generally represents the lower bound of performance across various scenarios, whereas the Cumulative method consistently reaches the upper bound of performance in all scenarios, consistent with the discussion in Section V. It is also apparent that the order of tasks affects the results; different sequences of tasks lead to variations in overall performance and forgetting in CL.

B. VAE VS TVAE

During the model training phase for Task 1, the generator is trained using $\boldsymbol{x}$, while the solver (OWD model) is trained with $(\boldsymbol{x}, y)$. To compare VAE and TVAE, we generate $\boldsymbol{x}'$ for Task 1 using both VAE and TVAE. As shown in Fig. 5, we then apply principal component analysis (PCA) with 2 principal components on the generated $\boldsymbol{x}'$ and visualize these components using a kernel density estimate (KDE) plot. Additionally, we plot the histogram of the corresponding y' by applying the OWD model to the generated $\boldsymbol{x}'$. We take the CL task sequence with Case ID 1 as an example. As demonstrated in Fig. 5, the distribution of $\boldsymbol{x}'$ and y' generated by TVAE closely resembles the real $(\boldsymbol{x}, y)$ of Task 1, compared to that generated by VAE. This indicates that TVAE is more effective in generating data samples that better simulate real samples, which allows SingleGen-TVAE to provide superior data for replay during CL model training, resulting in improved performance in OWD prediction compared to SingleGen-VAE.

C. SINGLE GENERATOR VS MULTI-GENERATOR

In the model training phase before training Task 12, the generator or generators is/are trained with $\boldsymbol{x}$ of Task 1 to Task 11, and the solver (OWD model) is trained with $(\boldsymbol{x}, y)$ of Task 1 to Task 11. To compare single generator and multi-generator, we generate $\boldsymbol{x}'$ of Task 1 to Task 11 by single generator and multi-generator. As shown in Fig. 6, we then perform PCA with 2 principal components on generated $\boldsymbol{x}'$ and visualize these components using the KDE plot. Additionally, we plot the histogram of the corresponding y' by applying OWD model on generated $\boldsymbol{x}'$. We take the CL task sequence with Case ID 1 as an example. As demonstrated in Fig. 6, with a focus on the main and tail parts of the histogram of y', the data distribution of y' generated by the multi-generator more closely resembles the real y compared to that generated by the single generator. As multi-generator could generate data samples that simulate the real samples better, MultiGen-TVAE has better generated data samples to replay for training of CL model, resulting in enhanced performance in OWD prediction compared to SingleGen-TVAE.

D. EVALUATION ON OWD TAIL

In addition to assessing OWD prediction across all data samples, it is important to evaluate the tail of the OWD distribution. To do this, we set the threshold and select the

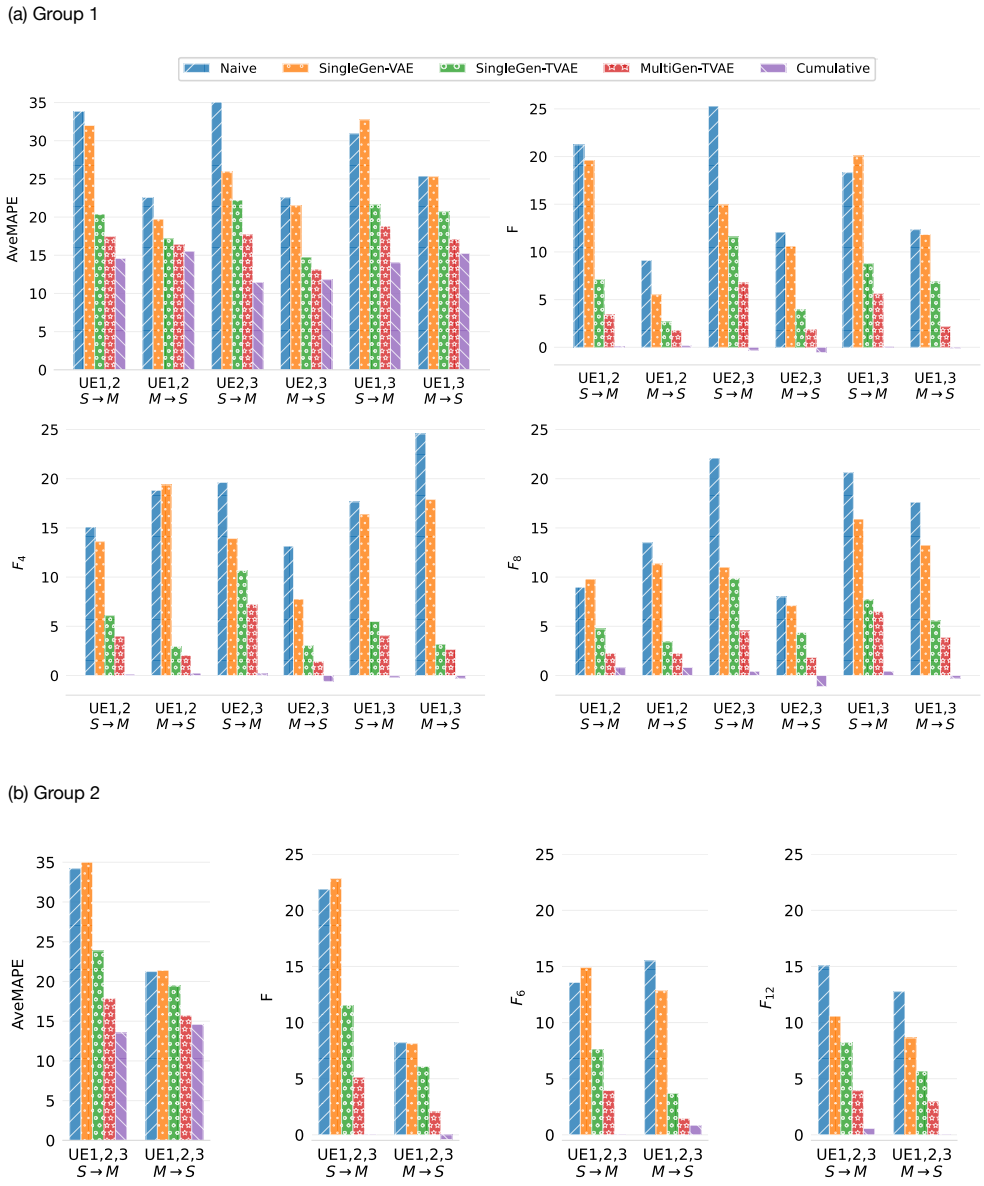

FIGURE 4. **Performance evaluation on CL task sequence Group 1 and Group 2.**

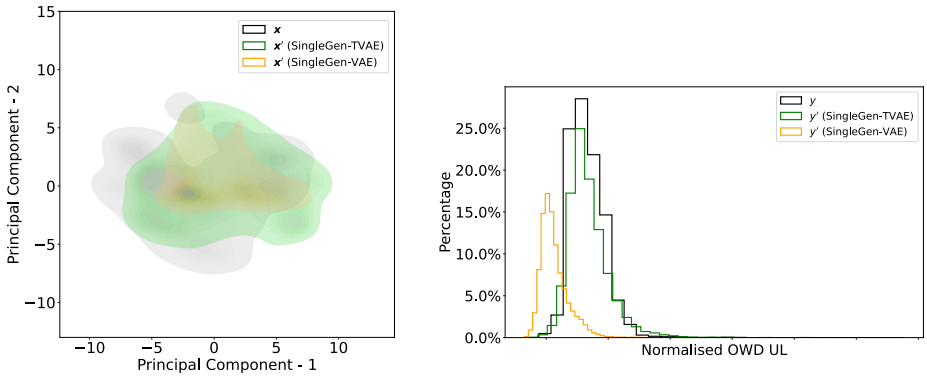

FIGURE 5. **KDE plot of principal components on generated x' by TVAE and VAE and histogram on corresponding y'.**

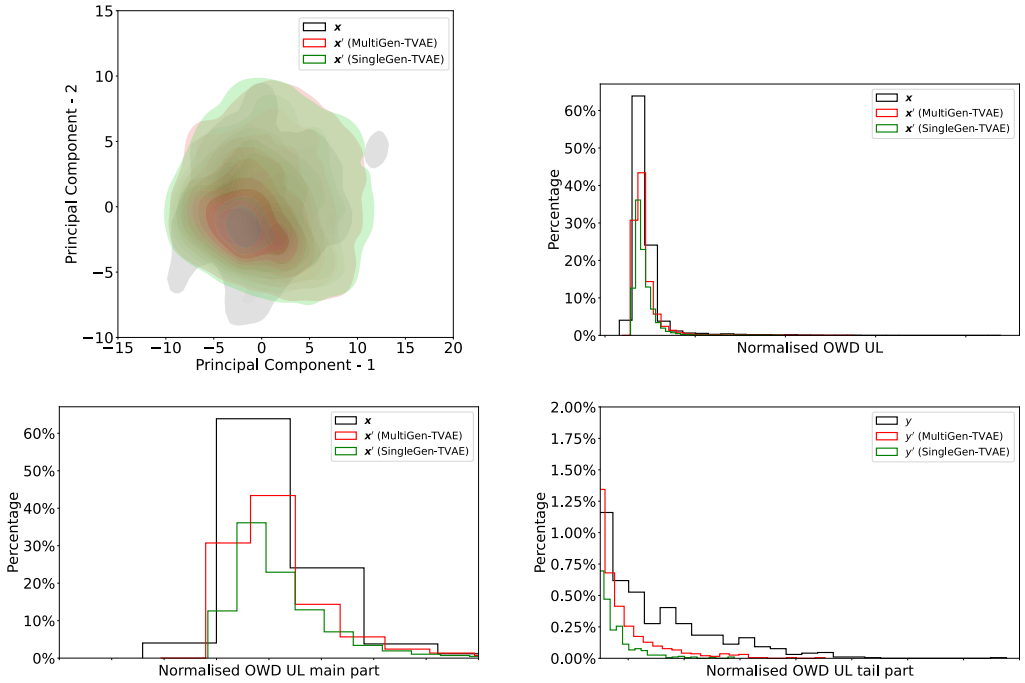

FIGURE 6. **KDE plot of principal components on generated x' by multi-generator and single generator and histogram on corresponding y', and zoom in on main part and tail part.**

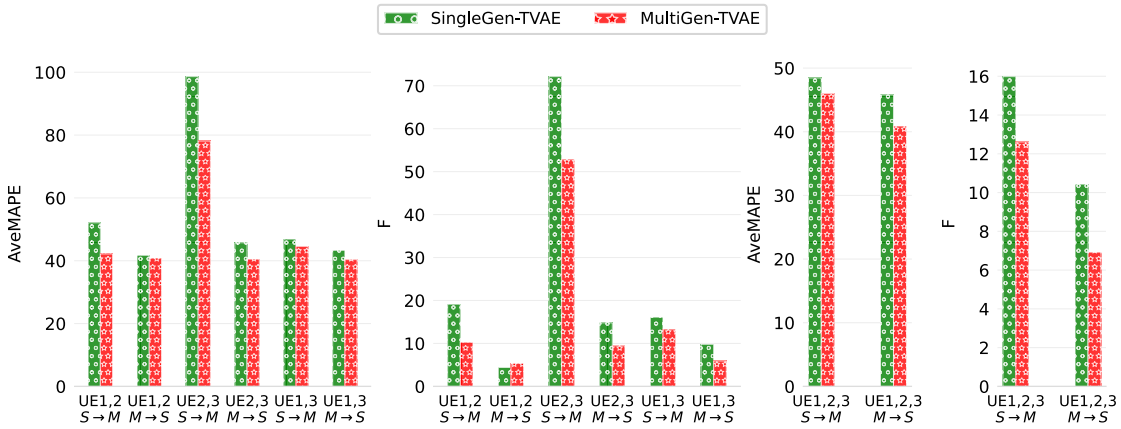

FIGURE 7. **Performance evaluation on OWD tail in CL task sequence Group 1 and Group 2 by metrics AveMAPE, F.**

data samples with OWD values larger than it, then calculate the evaluation metrics using only these selected samples. Fig. 7 presents the evaluation results for the tail of the OWD distribution using metrics AveMAPE and F with CL task sequences Group 1 and Group 2. The results indicate that, for nearly all task sequences, the proposed MultiGen-TVAE method outperforms SingleGen-TVAE in terms of tail performance for both AveMAPE and F metrics. This observation aligns with the findings in Section VI C that multi-generator could generate data samples that simulate the real samples better compared with single generator for OWD tail part.

E. COMPARISON OF GENERATORS WITH SAME SIZE

To compare the performance of SingleGen-TVAE and MultiGen-TVAE with generative models having an equivalent number of learnable parameters, we doubled the parameters for SingleGen-TVAE, as outlined in Table 3. Specifically, we increased each hidden layer in SingleGen-TVAE from 128 to 256 units. The evaluation was conducted using the CL sequence, with Case ID 3 as an example. As illustrated in Fig. 8, MultiGen-TVAE outperforms SingleGen-TVAE, despite the doubling of generator parameters in SingleGen-TVAE.

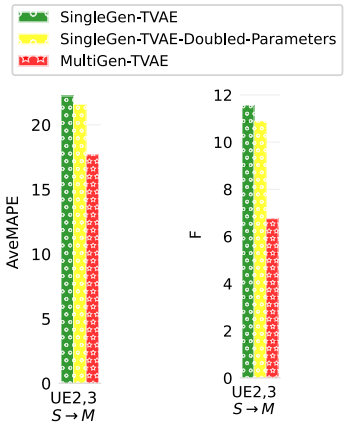

FIGURE 8. Performance evaluation on OWD by metrics AveMAPE, F with doubled size of generator parameters.

VII. DISCUSSION

In this section, we discuss the approach presented in this paper from both a methodological perspective and a use case perspective.

A. MULTI-GENERATOR APPROACH TO CL

From a methodological perspective, in the context of OWD prediction, we have made several key contributions aimed at mitigating the risk of catastrophic forgetting in CL. Firstly, we emphasized the significance of utilizing a tailored generative model to ensure that the data synthesis process aligns with the structural characteristics and specific requirements of the data. Given our use case involving tabular data, we adopted a generative model specifically designed for this type of data, the TVAE. This choice was shown to significantly enhance the quality of generated samples needed for effective replay and the retention of learned information. Next, we designed a multi-generator architecture for generative replay to tackle the challenges posed by single-generator systems, particularly in the presence of concept drift within the CL framework. By employing multiple generators, it was shown that we can reduce the effects of concept drift since each generator would be able to concentrate on producing samples tailored to a distinct concept, thereby enhancing the preservation of task-specific knowledge. Finally, to address the challenge of generator selection within our multi-generator generative replay framework, we implemented a domain-guided selection mechanism that utilizes domain knowledge for optimal task assignment to generators. In our use case, the differentiating factor for generator selection was the UE device type, provided as domain knowledge. This is one of the potential task configuration options and that other possibilities could be studied in future works. Moreover, moving forward, exploring data-driven approaches for generator selection presents a promising direction for future research. Through comprehensive empirical experiments, we demonstrated that the steps above significantly reduce risks of catastrophic forgetting and enhance the model's capability to sustain performance across a diverse array of tasks. By maintaining high-quality data replay and strategic task allocation, our approach enhances adaptability in dynamic environments.

Replay has been shown to be an effective strategy in CL if performance is the main objective. However, it requires large memory and often infeasible in real world applications where the access to past data is limited due to privacy-preserving. Instead of saving raw data, generative replay is a competitive alternative approach. As an example, the size of the raw data of task sequence with Case ID 1 is 170 MB. The size of one TVAE generator is 2.4 MB, and multi-generator with two TVAE generator in this case is 4.8 MB. When the number of tasks increase, the size of raw data increases linearly, while the size of generator won't change. This illustrates the benefits of our approach from an energy perspective. Our results show that the current single generator approach does not work well, in order to make it comparable to the multi-generator approach taken here it is reasonable to expect that it would get very costly from an energy perspective as this would require more training data and model parameters in the generator. For future work, it would be interesting to look closer at the generator training; how much energy does the training of the generator take compared to Replay-Based approaches with sample selection, and how large in terms of data, parameters and training would a successful single generator approach be. Furthermore, studying the energy consumption of this method compared to simple baselines such as isolated training for each task would be interesting.

B. ONE-WAY DELAY PREDICTION IN THE 3GPP ARCHITECTURE

We believe that the proposed approach for predicting OWD is a key enabler for performance verification, service assurance, and resource optimization in future networks. However, to fully exploit the opportunities, the approach must also align with the network design; specifically, the 3GPP architecture. In the following, we discuss this topic and present one possible perspective on how this integration can be realized[1].

Fig. 9 illustrates a network infrastructure having three main components: the RAN, the core network (CN), and OAM functionality. A set of UEs communicate with applications residing on the far side of the network. We envision the OWD model operating as part of the NWDAF in the core network, basing its predictions on baseband data streamed from the gNB via the baseband (BB) data inference pipeline. The OWD predictions can then be communicated back to the gNB for resource optimization, or to the application via the Network Exposure Function (NEF) to enable communication optimization at the application level.

As highlighted in this paper, the OWD model must be continuously updated to maintain its predictive performance.

[1]Note that this view does not necessarily reflect the opinions of Ericsson or our project partners.

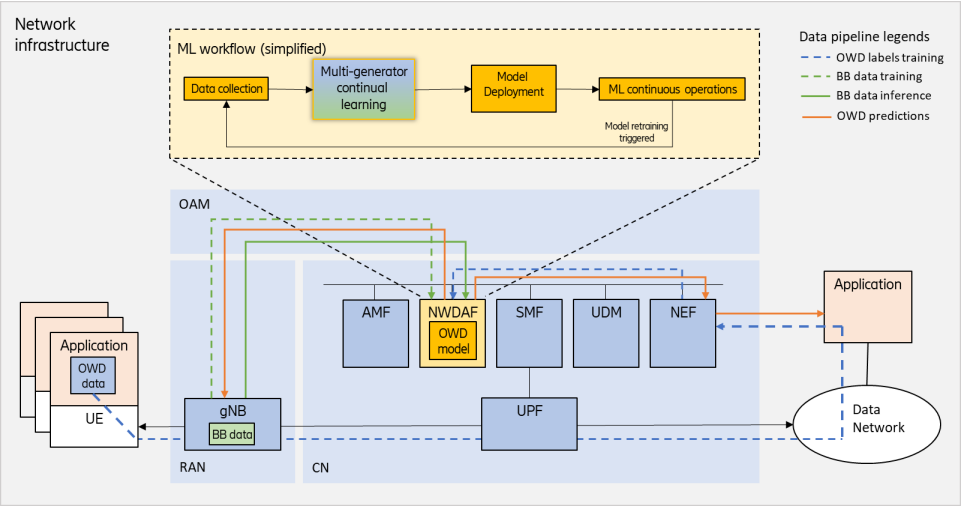

FIGURE 9. **The multi-generator CL approach as part of the ML workflow in NWDAF. Data pipelines are illustrative, and indicate that features and labels are collected differently depending on model inference or model retraining.**

This is managed within the NWDAF as part of the ML pipeline [15], responsible for continuously monitoring the performance of ML models in the NWDAF, and if deviations are detected, retraining is triggered. In such cases, our proposed approach is activated. During training, data is collected from UEs and gNBs using the dashed data pipelines.

Note that collecting OWD labels in operational networks may be challenging. It may not be feasible for all UEs and will likely depend on a subset of specifically instrumented UEs or controlled field trials. The exact process for label collection is outside the scope of this paper.

VIII. RELATED WORK

CL methods have been proposed to improve various aspects of ML and has been categorized in different ways [29] [30] [31]. In [29], based on how task specific information is stored and used throughout the sequential learning process, the authors distinguish five major categories, which are Regularization-Based approach, Replay-Based approach, Optimization-Based approach, Representation-Based approach and Architecture-Based approach. The different categories come with different properties related to energy usage as described in [32], where the authors compare different CL approaches in terms of energy consumption. Here the Representation-Based approaches score highest on energy efficiency but other approaches such as Replay-Based still outperform the naive baseline of joint training.

The focus of our paper is generative replay or pseudo-rehearsal, which is a sub-direction in the Replay-Based approach. Instead of storing old training samples, generative replay requires training a generative model to replay generated data. Compared with other approaches, one benefit of generative replay is that the generative model makes it possible to provide data samples from previous tasks for future needs, with much less memory and feasible for privacy concerns. As it is presented in Section III, DGR [19] establishes a framework that the learning of each new task is coupled with replaying generated data sampled from one generative model, ensuring the retention of previously acquired knowledge. MeRGAN [33] incorporates a memory replay mechanism to prevent catastrophic forgetting, which is further enforces through either joint training or replay alignment.

CL approaches in various categories have advantages and work best within specific scenarios. Between the categories, approaches are often orthogonal with respect to each other. Hence, generative replay could be hybrid with other CL strategies. To mitigate catastrophic forgetting of generative models, VCL [34] uses a weight regularization based variational Bayesian approach, and maintaining a distribution over parameters and updating it using evidence lower bound (ELBO). DGM [35] is inspired by biological synaptic plasticity principles, incorporating mechanisms like memory consolidation and task-specific adaptations.

For pseudo-rehearsal, the generative models could be of various types, such as generative adversarial networks (GANs) and variational autoencoder (VAE). GANs are typically used in scenarios where high-quality, realistic samples are required, such as image generation, super-resolution, and artificial data generation [35]. GANs are highly flexible and can be adapted to various types of generative tasks, but can be harder to control due to the adversarial nature. VAEs are more suited for tasks where a structured latent space and stable training are important [36] [37]. Since VAEs have an explicit, interpretable latent space, they are often more useful in applications where understanding the latent factors of data is important. Besides GANs and VAE, the novel approach DDGR [38] adopts a diffusion model as the generator and calculates an instruction-operator through the classifier to instruct the generation of samples.

IX. CONCLUSION

In this paper, we present a novel approach which introduces the concept of multi-generator for the state-of-the-art CL generative replay framework, along with TVAE as generative models. We emphasized the significance of utilizing a tailored generative model and producing samples tailored to a distinct concept by multi-generator, to enhance the preservation of data and task-specific knowledge. For our use case, the domain knowledge of UE capabilities is incorporated into the learning process for determining generator setup and relevance. The proposed approach is evaluated across a diverse set of scenarios with data that is collected in a realistic 5G testbed, demonstrates mitigation of catastrophic forgetting of OWD prediction and tail prediction model, in comparison to baselines. Strategic task allocation combined with high-quality data replay empowers our approach to adapt more effectively in dynamic environments. Furthermore, this approach reduces the need for data storage efficiently, addressing the challenges of resource constraints in 5G networks. Last but not least, we discuss how an OWD prediction engine can be integrated into the NWDAF in 3GPP architecture, which helps the application of the approach in operational networks.

For future research, one promising direction is to explore data-driven approaches for generator selection, and to evaluate the performance in comparison to domain-guided selection. In addition, future work may benefit from investigate alternative generative model architectures, such as temporal generative models, especially when OWD datasets are represented as time series data.

ACKNOWLEDGMENT

This research was supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) via the project Performance Prediction for Dependable 6G Networks through Causal Artificial Intelligence (2024-02438). This work is Co-funded by the European Union under Grant Agreement 101191936. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of all SUSTAIN-6G consortium parties nor those of the European Union or the SNS-JU (granting authority). Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] A. Karapantelakis and et al, "Co-creating a cyber-physical world," *Ericsson AB, White Paper*, 2024.

[2] Y. Liu, Y. Peng, B. Wang, S. Yao, and Z. Liu, "Review on cyber-physical systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 1, pp. 27–40, 2017.

[3] G. Almes, S. Kalidindi, M. J. Zekauskas, and A. Morton, "A One-Way Delay Metric for IP Performance Metrics (IPPM)," RFC 7679, 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7679

[4] S. Kalidindi, M. J. Zekauskas, and D. G. T. Almes, "A Round-trip Delay Metric for IPPM," RFC 2681, 1999. [Online]. Available: https://www.rfc-editor.org/info/rfc2681

[5] J. Ansari, C. Andersson, P. de Bruin, J. Farkas, L. Grosjean, J. Sachs, J. Torsner, B. Varga, D. Harutyunyan, N. König *et al.*, "Performance of 5G Trials for Industrial Automation," *Electronics*, vol. 11, no. 3, p. 412, 2022.

[6] J. Riihijarvi and P. Mahonen, "Machine learning for performance prediction in mobile cellular networks," *IEEE Computational Intelligence Magazine*, vol. 13, no. 1, pp. 51–60, 2018.

[7] S. Mostafavi, G. P. Sharma, and J. Gross, "Data-driven latency probability prediction for wireless networks: Focusing on tail probabilities," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 4338–4344.

[8] A. Rao, W. Tärneberg, E. Fitzgerald, L. Corneo, A. Zavodovski, O. Rai, S. Johansson, V. Berggren, H. Riaz, C. Kilinc *et al.*, "Prediction and exposure of delays from a base station perspective in 5G and beyond networks," in *Proceedings of the ACM SIGCOMM Workshop on 5G and Beyond Network Measurements, Modeling, and Use Cases*, 2022, pp. 8–14.

[9] C. Flinta, W. Yan, and A. Johnsson, "Predicting Round-Trip Time Distributions in IoT Systems using Histogram Estimators," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[10] F. S. Samani, R. Stadler, C. Flinta, and A. Johnsson, "Conditional density estimation of service metrics for networked services," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 2350–2364, 2021.

[11] A. Rao, H. Riaz, A. Zavodovski, R. Mochaourab, V. Berggren, and A. Johnsson, "Generalizable One-Way Delay Prediction Models for Heterogeneous UEs in 5G Networks," in *NOMS 2024-2024 IEEE Network Operations and Management Symposium*. IEEE, 2024, pp. 1–9.

[12] H. Larsson, F. Moradi, J. Taghia, X. Lan, and A. Johnsson, "Domain adaptation for network performance modeling with and without labeled data," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2023, pp. 1–9.

[13] A. Rao and M. Boman, "Self-supervised Pretraining for User Performance Prediction under Scarce Data Conditions," *Authorea Preprints*, 2025.

[14] F. Moradi, R. Stadler, and A. Johnsson, "Performance prediction in dynamic clouds using transfer learning," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 242–250.

[15] 3GPP, "Architecture Enhancements for 5G System (5GS) to Support Network Data Analytics Services," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.288, 2024, version 19.1.0.

[16] M. Polese, L. Bonati, S. D'oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[17] S. Dohare, J. F. Hernandez-Garcia, Q. Lan, P. Rahman, A. R. Mahmood, and R. S. Sutton, "Loss of plasticity in deep continual learning," *Nature*, vol. 632, no. 8026, pp. 768–774, 2024.

[18] M. McCloskey and N. J. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem," *Psychology of Learning and Motivation*, vol. 24, pp. 109–165, 1989.

[19] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual Learning with Deep Generative Replay," in *Neural Information Processing Systems*, 2017.

[20] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling Tabular data using Conditional GAN," in *Neural Information Processing Systems*, 2019.

[21] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *CoRR*, vol. abs/1312.6114, 2013.

[22] K. Hedayat, R. Krzanowski, A. Morton, K. Yum, and J. Babiarz, "A Two-Way Active Measurement Protocol (TWAMP)", RFC 5357," 2008.

[23] 3GPP, "NR; Physical layer procedures for data," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.214, 2021, version 16.5.0.

[24] ——, "NR; Physical layer measurements," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.215, 2021, version 16.5.0.

[25] ——, "NR; Medium Access Control (MAC) protocol specification ," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.321, 2021, version 16.5.0.

[26] N. Patki, R. Wedge, and K. Veeramachaneni, "The Synthetic data vault," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Oct 2016, pp. 399–410.
[27] D. Maltoni and V. Lomonaco, "Continuous Learning in Single-Incremental-Task Scenarios," *Neural networks : the official journal of the International Neural Network Society*, vol. 116, pp. 56–73, 2018.
[28] D. Lopez-Paz and M. Ranzato, "Gradient Episodic Memory for Continual Learning," in *Neural Information Processing Systems*, 2017.
[29] L. Wang, X. Zhang, H. Su, and J. Zhu, "A Comprehensive Survey of Continual Learning: Theory, Method and Application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, pp. 5362–5383, 2023.
[30] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges," *Information Fusion*, vol. 58, pp. 52–68, 2020.
[31] M. D. Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. G. Slabaugh, and T. Tuytelaars, "A Continual Learning Survey: Defying Forgetting in Classification Tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 3366–3385, 2019.
[32] T. Trinci, S. Magistri, R. Verdecchia, and A. D. Bagdanov, "How green is continual learning, really? Analyzing the energy consumption in continual training of vision foundation models," *arXiv preprint arXiv:2409.18664*, 2024.
[33] C. Wu, L. Herranz, X. Liu, Y. Wang, J. van de Weijer, and B. Raducanu, "Memory Replay GANs: learning to generate images from new categories without forgetting," in *Neural Information Processing Systems*, 2018.
[34] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational Continual Learning," in *International Conference on Learning Representations*, 2018.
[35] O. Ostapenko, M. M. Puscas, T. Klein, P. Jähnichen, and M. Nabi, "Learning to Remember: A Synaptic Plasticity Driven Framework for Continual Learning," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11 313–11 321, 2019.
[36] R. Kemker and C. Kanan, "FearNet: Brain-Inspired Model for Incremental Learning," in *International Conference on Learning Representations*, 2018.
[37] M. Riemer, T. Klinger, D. Bouneffouf, and M. M. Franceschini, "Scalable Recollections for Continual Lifelong Learning," in *AAAI Conference on Artificial Intelligence*, 2017.
[38] R. Gao and W. Liu, "DDGR: Continual Learning with Deep Diffusion-based Generative Replay," in *International Conference on Machine Learning*, 2023.