Towards Lifelong Unseen Task Processing with a Lightweight Unlabeled Data Schema for AIoT

Tianyu Tu, Zhili He, Zhigao Zheng, *Member, IEEE,* Zimu Zheng, *Member, IEEE,* Jiawei Jiang, *Member, IEEE,* Yili Gong, *Member, IEEE,* Chuang Hu, *Member, IEEE,* and Dazhao Cheng, *Senior Member, IEEE*

***Abstract*—With the rapid development of the Internet of Things (IoT), IoT devices find applications in various domains. The data generated by these devices is utilized for analysis and services, especially in the field of Artificial Intelligence (AI) applied to IoT, known as Artificial Intelligence of Things (AIoT). The enhancement of edge device computing power in the IoT has led to the emergence of research areas like edge-cloud synergy AI theories and application services. In the context of lifelong learning and real-time processes in AIoT edge-cloud synergy services, addressing unseen tasks becomes crucial. Unseen tasks arise when inference requests from edge devices involve models not present in the cloud's model repository. Addressing these challenges involves generating data to either augment small sample problems or alter the data distribution for heterogeneous sample issues. As the application of large language models (LLMs) for data generation gains traction, challenges emerge in the context of AIoT edge-cloud synergy services. Firstly, fine-tuning LLMs with heterogeneous data exacerbates model bias issues. Secondly, the substantial data requirements for training LLMs pose a contradiction. Lastly, the involvement of manual annotation in LLM-based data generation introduces complexity and cost. This paper proposes a framework Seafarer to these challenges using Generative Adversarial Networks and Self-taught Learning. Seafarer avoids model bias, reduces data requirements, and eliminates the need for manual annotation. The design demonstrates effectiveness theoretically and is validated on the Cityscapes dataset, achieving an 80% reduction in training loss and improved validation loss stability.**

***Index Terms*—Edge-Cloud Synergy AI, Lifelong Learning, Internet of Things (IoT), Artificial Intelligence of Things (AIoT).**

I. Introduction

With the rapid development of the Internet of Things (IoT), IoT devices have been applied in various fields. The data generated by IoT devices is utilized for analysis and services, especially in the field of Artificial Intelligence (AI). The integration of AI into the paradigm of the IoT is known as Artificial Intelligence of Things (AIoT), such as Intelligent Medical [1], Smart City [2], and Smart Farming [3]. Simultaneously, as the computational capabilities of IoT edge devices are enhanced [4], [5], [6], research focus has shifted towards the theoretical frameworks of edge-cloud synergy AI based on AIoT (*e.g.* federated learning [7] and edge-cloud collaborative inference [8]) and application services (*e.g.* KubeEdge-Sedna [9]).

The theoretical frameworks and services of edge-cloud synergy AI based on AIoT leverage data collected from IoT devices such as robots [10], monitors [11], and sensors [12], and these data are utilized on servers to train models. In practice, deploying edge-cloud synergy AI services is a lifelong-learning and real-time process. For instance, the KubeEdge-Sedna edge-cloud synergy AI service on the cloud continuously trains, updates, and stores models in real-time using data collected at the IoT edge. The edge then sends requests to the cloud for real-time inference based on the trained models. In the thermal comfort prediction application [13] of KubeEdge-Sedna, lifelong learning is achieved by accumulating metaknowledge, which captures information about the model's performance on various tasks and metadata (attributes used to describe datasets, such as season, dataset names, data columns, or data subsets). A metaknowledge base enables the model to adapt to new tasks, retain previous knowledge, and make informed decisions regarding task-model selection. By leveraging metaknowledge, the model can enhance its learning capabilities, surpass non-lifelong learning methods, and maintain a leading position in thermal comfort prediction.

In the lifelong learning process of edge-cloud synergy AI services based on AIoT, inference requests sent from the IoT edge pertain to tasks that are not present in the cloud's model repository. In other words, edge-cloud synergy AI services encounter unseen tasks during their running [13], [14]. Faced with these unseen tasks, the cloud server needs to promptly utilize the data collected at the edge to train a new model. However, due to the mobility of IoT devices and user characteristics, the data collected at the IoT edge devices may be limited in sample size, making it insufficient for model training [15]. Additionally, these data may be heterogeneous samples (*i.e.*, non-independently and identically distributed (non-IID)), leading to biased models during training [16].

Intuitively, the key to addressing unseen tasks is data generation. This involves either increasing data examples based on existing data (addressing the small sample problem) or altering the data distribution (addressing the heterogeneous sample problem). With the deepening of research on Large Language Models (LLMs) [17], [18], [19], using the data generation capability of these models to tackle unseen tasks during the lifelong learning process of edge-cloud synergy AI services is a potential approach. However, the application of

Tianyu Tu, Zhili He, Zhigao Zheng, Jiawei Jiang, Yili Gong, Chuang Hu, and Dazhao Cheng are with the Computer School, Wuhan University, Hubei 430072, China. E-mail: {meetnailtu30, 2022182110069, zhengzhigao, jiawei.jiang, yiligong, handc, dcheng}@whu.edu.cn. *(Corresponding authors: Yili Gong; Zhigao Zheng.)*

Zimu Zheng is with the Edge Cloud Innovation Lab, Huawei Cloud. Email: zimu.zheng@huawei.com.

TABLE I
MULTILINGUAL UNDERSTANDING OF GEMINI ULTRA.

Gemimi Ultra	English	French	Hindi	Chinese
XM-3600 (CIDER)	86.4	77.9	***31.1***	33.3

LLMs in edge-cloud synergy AI services based on AIoT faces three major challenges.

Firstly, fine-tuning LLMs with heterogeneous data exacerbates the problem of model bias. An intuitive approach is to fine-tune a pretrained large language model with data collected at the IoT edge devices, enabling it to generate specific data to address unseen tasks. However, this practice leads to bias issues in the fine-tuned large language model. For example, as shown in Table I, the Gemini large language model [20] exhibits weak generation capabilities for the Hindi language. Updating this model with collected heterogeneous data would further accentuate the bias problem.

Secondly, the training requirements for LLMs are substantial. Taking Gemini as an example, model training often involves a large number of parameters (Gemini Nano-1 has 180 million parameters, and Gemini Nano-2 has 325 million parameters). Opting to retrain LLMs for data generation introduces a contradiction. Due to the substantial parameter size of LLMs, the data requirements for training them are high. This creates a paradox, as the existence of a large amount of data would seemingly eliminate the issue of small sample size.

Thirdly, the data generation process based on LLMs involves a manual annotation process [21]. The commands given to LLMs for data generation often include information such as locations, objects, and events, all of which require human judgment for annotation. Manual annotation is a complex and costly process. The data collected at the IoT edge devices is often unlabeled, and when faced with such unlabeled data, LLMs cannot directly generate more data based on it.

In this paper, we propose utilizing Generative Adversarial Networks (GANs) and Self-taught Learning (STL) to address the aforementioned three challenges, and design a framework Seafarer for resolving unseen tasks during the lifelong learning process of edge-cloud synergy AI services based on AIoT. We summarize our contributions as follows:

Firstly, Seafarer avoids falling into the issue of model bias. Unlike the approach of fine-tuning LLMs with heterogeneous data to address unseen tasks, Seafarer utilizes GANs to generate data, solving small sample and heterogeneous sample problems. Through STL, we extract data representations to reduce data dimensionality, which are then provided for use in addressing unseen tasks. The models for unseen tasks receive incremental and non-heterogeneous data representations, ensuring that training and updating for unseen tasks do not succumb to data bias issues.

Secondly, Seafarer has a low requirement for training data. The GANs we use are designed for small sample data, preventing overfitting while extracting more features through deepening the network layers. Additionally, the network design for STL does not require a large number of parameters. In contrast to LLMs that need a considerable number of parameters for generalization, our design addresses unseen tasks with an appropriate parameter size.

Thirdly, Seafarer does not require manual annotation. Our training process relies entirely on unlabeled data. Initially, we use GANs to learn the distribution of small sample or heterogeneous sample data. Subsequently, we generate a substantial amount of unlabeled data using the trained GANs. Finally, we employ STL to extract data representations that reduce data dimensionality, facilitating easier training for unseen tasks.

By combining GANs and STL, Seafarer has been theoretically proven to be effective. Additionally, we validate the effectiveness of Seafarer over the Cityscapes dataset [22], achieving an 80% reduction in training loss and greater stability in validation loss, compared to the training without our added modules.

In the remainder of the paper, we first provide background information in Section II. We propose the system architecture in Section III. Following that, in Section IV, we define the problems to be addressed. Subsequently, we showcase our design and algorithms in Section V. Moving on, in Section VI, we present the specific implementation of our design. Besides, In Section VII, we discuss experiments and results. Finally, Section IX presents the conclusion and outlines future work.

II. PRELIMINARIES AND BACKGROUND

A. Edge-cloud Synergy AI Based on AIoT

With the enhancement of computational capabilities in IoT edge devices, edge-cloud synergy AI based on AIoT has gradually become a research focus. Edge-cloud synergy AI aims to package mainstream research outcomes (such as Joint Inference [23], Incremental Training [24], and Federated Learning [7], etc) into services for rapid user adoption. For lifelong learning process, the edge-cloud synergy AI service maintains a knowledge repository in the cloud, used for storing historically trained models. IoT edge devices send collected data to the cloud for model updating and training. When faced with inference tasks, IoT edge devices request the corresponding models from the cloud based on target task information, *i.e.*, metaknowledge.

In this paper, contexts are partitioned as *tasks* in the application of edge-cloud synergy AI based on AIoT. Each task is associated with a specific model. Similar to [25], we describe a task as a series of model training or inference processes with metaknowledge and IoT edge device characteristics. We adopt the definitions of metaknowledge, lifelong learning, and unseen tasks from [13].

Metaknowledge (*i.e.*, knowledge of knowledge) is a mapping between given arbitrary metadata and the performance descriptor of a metamodel on this metadata. In the edge-cloud synergy AI, metaknowledge intrinsically is the knowledge about a model that can be used to guide the usage and maintenance of this model. Formally,

Definition 1. *(Metaknowledge). The metaknowledge $f_j^{[n]}(\cdot)$ of jth task is*

$$f_j^{[n]}(\cdot) : x_j^{[n]} \rightarrow y_j^{[n]}, \tag{1}$$

where $x_j^{[n]} = [x_t^{[n]}]_j$ is the n-order metadata of the tth sample $s_j = [s_t]_j$ and n is the number of order. $y_j^{[n]} = [y_t^{[n]}]_j$ is the n-order descriptor of s_j in task J_j.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3396282

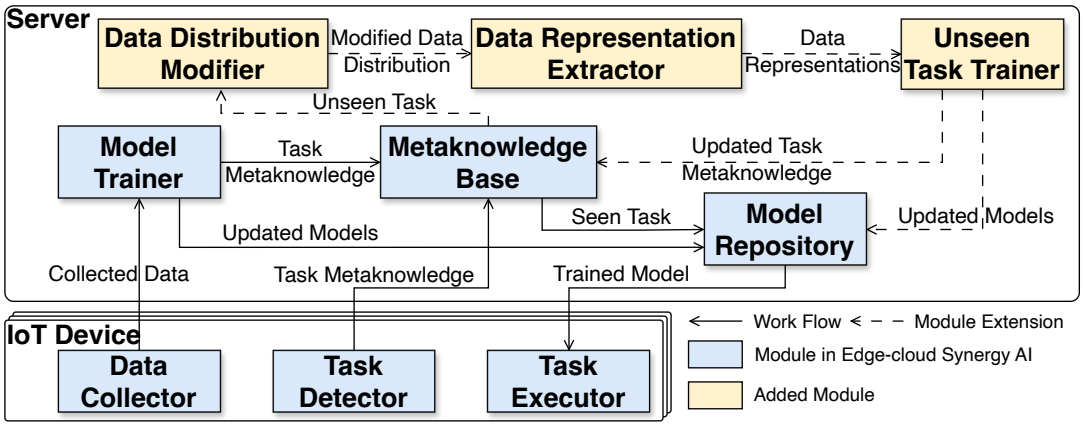

Fig. 1. Proposed Seafarer framework architecture includes three modules: **Data Distribution Modifier**, **Data Representation Extractor**, and **Unseen Task Trainer**. The **Data Distribution Modifier** module alters sample distribution using GANs. The **Data Representation Extractor** module extracts data representations via STL. The **Unseen Task Trainer** module trains unseen tasks using the data representation dataset.

Confronted with the issues of heterogeneous data distributions and small samples on the wide variety of IoT edge, lifelong learning is necessary for edge-cloud synergy AI. The edge-cloud synergy lifelong learning learns unseen tasks with shared knowledge among various scenarios over time, and leverages the cloud knowledge base, empowering the scheme with memory ability, which helps to continuously learn and accumulate historical knowledge to overcome the catastrophic forgetting challenge. Formally,

Definition 2. *(Lifelong Learning). Given historical tasks at past N time slots $\{J_{T-N}, ..., J_{T-1}\}$, the goal of lifelong learning is to minimize the prediction error of the future coming tasks $\{J_T, ..., J_{T+M}\}$, i.e.,* $\min \sum_{j\in\mathcal{J}} \sum_{t\in t_j} \|f_j([x_t]_j) - [y_t]_j\|_2^2$.

Note that the coming tasks $\{J_T, ..., J_{T+M}\}$ are not necessarily in the historical tasks $\{J_{T-N}, ..., J_{T-1}\}$ and any elements in a previous task can be updated in the future. In other words, the process of lifelong learning encounters unseen tasks. Formally, we define an unknown task as follows:

Definition 3. *(Unseen Task). A task is an unseen task if $\exists n, \delta^{[n]} > \epsilon^{[n]}$, where $f_j^{[n]}$ and $f_i^{[n]}$ are the metaknowledge of jth task in metaknowledge base and the ith inference task, respectively. $\delta^{[n]} = \|f_j^{[n]}, f_i^{[n]}\|$ is the metaknowledge similarity, $\epsilon^{[n]}$ is the similarity threshold.*

B. Tackling Unseen Tasks

The essence of unseen tasks is fundamentally a small (labeled) sample learning problem. In addition to the previously mentioned solutions of LLMs and their corresponding challenges, mainstream methods include the following.

Acquiring labeled data for machine learning is often challenging and expensive. Therefore, training models using unlabeled data and a small amount of labeled data is a practical and promising approach in real-world production. Existing methods for this purpose include semi-supervised learning [26], transfer learning [27], and self-taught learning [28].

Semi-supervised learning distinguishes between labeled and unlabeled data. For labeled points, the algorithm uses traditional supervision to update model weights, while for unlabeled points, the algorithm minimizes the prediction differences between similar training examples. It is worth noting that semi-supervised learning often makes additional assumptions, such as assuming that unlabeled data can be labeled with the same labels as the classification task, and these labels are merely unobserved. Therefore, semi-supervised learning is not suitable for the dynamic characteristics of target changes in edge-cloud synergy AI services.

Transfer learning typically requires more labeled data from different but related tasks, with its core involving transferring knowledge from one supervised learning task to another. Consequently, it requires additional labeled data for these other supervised learning tasks rather than unlabeled data. Therefore, under conditions of manual labor intensity and timely processing, transfer learning is not suitable for edge-cloud synergy AI services.

Self-taught learning (STL) does not assume that unlabeled data follows the same class labels or generative distribution as labeled data. It posits that even many randomly downloaded data may contain basic patterns (*e.g.*, edge information in image data) similar to the labeled small sample data. If one can learn to identify such patterns from unlabeled data beforehand, these patterns can be utilized for the target learning tasks. Specifically, self-taught learning employs unlabeled data to learn concise and higher-level data representations of inputs, making the interested classification tasks more accessible.

Due to the less restrictive requirements of STL on the types of unlabeled data, it is much easier to apply in many practical applications (such as image, audio, or text classification) compared to typical semi-supervised or transfer learning methods. Therefore, our design utilizes STL as the method for data representation extractor in the lifelong learning of edge-cloud synergy AI services based on AIoT.

III. System Architecture

The framework we design for lifelong learning in edge-cloud synergy AI services is illustrated in Fig. 1.

The **Data Collector** module on the IoT edge devices sends real-time collected data to the server. The **Model Trainer** module on the server utilizes the collected data for training, stores the training results in the **Model Repository**, and stores task metaknowledge in the **Metaknowledge Base**. The server's **Metaknowledge Base** compares the task metaknowledge sent by the edge's **Task Detector** module. If it is a seen task, the server sends the corresponding model from the **Model Repository** to the IoT edge device. The **Task Executor** module on the IoT edge devices executes specific inference tasks.

For detected unseen tasks, the **Data Distribution Modifier** module first learns the distribution of the collected small samples or heterogeneous samples from IoT edge devices. Subsequently, based on this learned data distribution, it modifies the quantity of small samples or the distribution of heterogeneous samples collected from IoT edge devices. This ensures that the sample quantity is sufficient for training unseen tasks and the sample distribution is independent and identically distributed (IID). Following that, the **Data Representation Extractor** module extracts the data basics of the modified IoT edge device data and represents the original small samples or heterogeneous data based on the extracted data basics. Finally, the **Unseen Task Trainer** module utilizes the extracted data representation and the original data labels to train unseen tasks. This not only accelerates training convergence but also enhances training accuracy.

We emphasize that we only added new modules (**Data Distribution Modifier** module, **Data Representation Extractor** module, and **Unseen Task Trainer** module) on the server side. The added new modules collaboratively solve unseen tasks during the lifelong learning process of edge-cloud synergy AI services based on AIoT with minimal cost, and without modifying the original edge-cloud synergy AI architecture.

Our design is suitable for general edge-cloud synergy AI services based on AIoT. At the same time, our design is lightweight compared to aforementioned LLMs solutions, and the process of our design does not require any sample label. Therefore, our design is a *lightweight unlabeled* data schema for solving unseen tasks during the lifelong learning process of edge-cloud synergy AI services based on AIoT.

IV. Problem Definition

For the lifelong learning of unseen tasks in edge-cloud synergy AI services, our design relies solely on data collected from IoT edge devices, eliminating the need for additional devices and the involvement of data scientists. This ensures that the service is timely and automatic at any time and location. The target of this paper is to achieve rapid training convergence for unseen tasks requested by any IoT edge device, constrained in scenarios where task similarity is low (*i.e.*, historical tasks cannot be leveraged), the quantity of samples collected at IoT edge devices is limited, or the sample distribution is heterogeneous. In what follows, we formally define notations and our problem.

Problem *Edge-cloud Synergy AI Lifelong Unseen Task Handling* (P1): Given samples $\{X_1, ..., X_N\}$ collected from N IoT edge devices, and the unseen task $f(w)$ request from any IoT edge device, the goal is to minimize the training loss of the unseen task with the minimum training time T, *i.e.*, the expected training loss converges to the minimum value F^* with a ρ precision with the minimum training time T:

$$\min \mathbf{E}(T(f(w))), \tag{2}$$

$$s.t. \quad \mathbf{E}(\phi_{I_t}(f(w); y)) - F^* \leq \rho, \tag{3}$$

$$\exists n, \delta^{[n]} > \epsilon^{[n]}, \tag{4}$$

$$\sum_{i=1}^{N} |X_i| < \tau, \tag{5}$$

$$\exists (i, j) \in N, supp(X_i) \cap supp(X_j) \to \emptyset, \tag{6}$$

where ϕ_{I_t} denotes the outcome of loss function ϕ over I_tth mini-batch sample, y denotes the sample labels, n denotes the metadata order of the sample, $\delta^{[n]}$ is the metaknowledge similarity, $\epsilon^{[n]}$ is the similarity threshold, and τ is the training quantity threshold.

Constraint Analysis. Constraint 3 is introduced to demonstrate that the training of unseen tasks can converge. Constraint 4 is aimed at highlighting the low metaknowledge similarity between the current task and historical tasks, indicating that leveraging historical tasks for the current task is not feasible. Constraint 5 emphasizes that the total quantity of samples collected from IoT edge devices is insufficient to train unseen tasks. Constraint 6 indicates that the support sets of IoT edge device data are disjoint, meaning that the data distribution is heterogeneous.

The expectation in $\mathbf{E}(T(f(w)))$ and $\mathbf{E}(\phi_{I_t}(f(w); y))$ is due to the randomness in IoT edge devices and SGD. Solving Problem **P1** is challenging in two aspects:

1) In the case of small sample or heterogeneous data X_i, training for unseen tasks $f(w)$ is prone to model bias. Therefore, it is essential to augment small samples, and modify data distribution for heterogeneous samples, correcting data quantity and distribution.
2) Generally, achieving rapid convergence without leveraging historical tasks is challenging. Therefore, it is necessary to reduce the dimensionality of the training data, accelerating training convergence $\mathbf{E}(\phi_{I_t}(f(w); y))$ while improving training accuracy.

In Section V-B and Section V-C, we address these two challenges, respectively, and propose approximate algorithms to find an approximate solution to Problem **P1** efficiently. In Section V-D, we prove our design theoretically.

V. Seafarer Deign

To address the aforementioned Problem **P1**, we present Seafarer, a framework for resolving unseen tasks during the lifelong learning process of edge-cloud synergy AI services based on AIoT, aiming to tackle issues arising from sample scarcity and heterogeneity.

A. Design Overview

The input of the framework for processing an unseen task consists of small sample data, or heterogeneous data, corresponding labels for the small sample data or heterogeneous

Algorithm 1: Training GANs in Lifelong Learning Edge-cloud Synergy AI Services

Input: The number of training epochs E, minibatch size m, number of steps to apply to the discriminator k.

Output: The trained GANs.

1 **for** $i = 1 : E$ **do**
2 **for** $s = 1 : k$ **do**
3 Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_g(z)$.
4 Sample minibatch of m examples $\{x^{(1)}, ..., x^{(m)}\}$ from data generating distribution $p_{data}(x)$.
5 Update the discriminator by ascending its stochastic gradient: $\nabla_{\theta_d} D_{loss}(\theta_d)$. // Eq. 8
6 **end**
7 Sample minibatch of m noise samples $\{z^{(1)}, ..., z^{(m)}\}$ from noise prior $p_g(z)$.
8 Update the generator by descending its stochastic gradient: $\nabla_{\theta_g} G_{loss}(\theta_g)$.// Eq. 9
9 **end**

data, and the model to be trained (*i.e.*, the unseen task). The execution process can be summarized as follows: 1) Train a GAN model based on the small sample data or heterogeneous sample; 2) Utilize the GAN model trained in Step 1 to generate a certain quantity of unlabeled data to modify the quantity of sample or the distribution of the sample; 3) The STL module extracts the basic data representations of the unlabeled data generated by the GAN; 4) The STL module represents the original small or heterogeneous sample data using the learned basic data representations; 5) Train the model using the extracted data representations and the original labels of the small or heterogeneous sample data; 6) Upon convergence of training, return the well trained model.

For an unseen task where existing labeled data samples are scarce or heterogeneous, not conforming to an IID distribution, we address the modification of data distribution in Section V-B. Regarding GANs, which produce unlabeled data, we extract representations of this unlabeled data in Section V-C to effectively utilize it. We prove the effectiveness of Seafarer theoretically in Section V-C.

B. Data Distribution Modifier

The purpose of the **Data Distribution Modifier** module is to alter the quantity or distribution of the original data collected from IoT edge devices, aiming to address the impact of the small sample problem, preventing the convergence of training for unseen tasks, or the impact of the heterogeneous sample problem, leading to model bias in the training of unseen tasks.

We employ GANs in the role of a data distribution modifier. GANs are predominant generative models widely employed in various domains such as image restoration [29], healthcare [30], and 3D printing [31]. GANs, due to their powerful capability to learn data distributions, are well-suited for assimilating information from IoT edge devices. However, attempting to train unseen tasks using existing small samples or heterogeneous data collected from IoT edge devices poses challenges. Typically, training GANs also requires a substantial amount of training data, making it challenging to address this issue.

To enable GANs to learn on small sample or heterogeneous data, the structure of the GAN needs to be specially designed. Therefore, we adopt the design approach outlined in [32]. Specifically, due to the limited number of samples, the generator G network layers are designed to be deep to fully utilize the information in the data. However, the depth of the network may lead to learning forgetting and degradation [33]. To address this, residual layers [34] are used to pass shallow-layer information to the deep network. The discriminator D network is treated as an encoder and is trained with small decoders. Such auto-encoding training forces discriminator D to extract data features that the decoders can give good reconstructions. The decoders are optimized together with discriminator D on a reconstruction loss L_{recons}, which is only trained on real samples:

$$\mathcal{L}_{recons} = \frac{1}{m}\sum_{i=1}^{m}[\|\mathcal{G}(f) - \mathcal{T}(x^{(i)})\|], \tag{7}$$

where f is the intermediate feature-maps from D, the function $\mathcal{G}$ contains the processing on f and the decoder, and the function $\mathcal{T}$ represents the processing on m data samples $\{x^{(i)}, ..., x^{(m)}\}$.

Therefore, we have the following loss function for the discriminator D:

$$\begin{aligned} D_{loss}(\theta_d) = &\frac{1}{m}\sum_{i=1}^{m}[\min(0, -1 + D(x^{(i)})) \\ &+ \min(0, -1 - D(G(z^{(i)})))] - Eq.7, \end{aligned} \tag{8}$$

while the generator remains the same as proposed in the original work [35]:

$$G_{loss}(\theta_g) = \frac{1}{m}\sum_{i=1}^{m}[D(G(z^{(i)}))]. \tag{9}$$

The training process of GANs in the lifelong learning process of edge-cloud synergy AI services based on AIoT follows the typical training flow of GANs, as depicted in Algorithm 1. It starts by training the discriminator D with m mini-batch of real data and m synthetic data generated by GANs iteratively (line 2-6). Then the generator G is trained over m noise samples with the help of the discriminator D (line 7-8). This process continues until the generator G converges or reaches a specified number of iterations E.

We emphasize that in the lifelong learning process, the dimensions of small or heterogeneous samples are unknown, and resizing samples to be either too large or too small adversely affects training [36]. Therefore, we augment the adopted architecture with options for different input feature dimensions and corresponding variations in the number of feature extraction layers. After collecting data from IoT edge devices, the server resizes the data to the most suitable dimensions based on its features, which are then used for

GANs training.

Addressing Heterogeneous Samples with Conditional GANs. It is observed that for scenarios with highly heterogeneous data distributions, generic solutions may prove ineffective. Conditional GANs [37], by introducing label embedding y with original unlabeled training samples, enhance the capability to generate data with specific distributions. Therefore, we use Conditional GANs to address this particular scenario. The loss function for the discriminator D is related to label embedding y, *i.e.*,

$$\begin{aligned} D_{loss}(\theta_d) = & \frac{1}{m}\sum_{i=1}^{m}[\min(0, -1 + D(x^{(i)}|y)) \\ & + \min(0, -1 - D(G(z^{(i)}|y)))] - Eq.7. \end{aligned} \tag{10}$$

The loss function for the generator G is also related to label embedding y, *i.e.*,

$$G_{loss}(\theta_g) = \frac{1}{m}\sum_{i=1}^{m}[D(G(z^{(i)}|y))]. \tag{11}$$

The use of Conditional GANs is not fundamentally different from the generic GANs. The only distinction is the introduction of label embedding during GAN training. All other steps remain consistent.

Modify Data Distribution. The trained GAN augments small samples to generate samples meeting the requirements τ for training unseen tasks, *i.e.*, $\sum_{i=1}^{N}|X_i| + \sum_{j=1}^{n}|G(z^{(j)})| \geq \tau$, where n is the quantity of data generated by GANs (Constraint 5). It also modifies the data distribution for heterogeneous samples to ensure that the sample distribution is IID, *i.e.*, $\forall(i,j) \in N, supp(X_i) \cap supp(X_j) \neq \emptyset$ (Constraint 6).

Reuse Historical Data. We provide two modes of data reuse. For servers with good inference performance, we do not store historical data, but rather regenerate data each time using GANs. However, in the case of cloud servers with poor inference performance, the data generation time of GANs increases, which cannot be overlooked. To address this issue, we store the data in the form of key-value pairs, with metadata as the key and the data generated by GANs as the value. When encountering tasks with different metaknowledge but similar metadata, historical data can be reused.

C. Data Representation Extractor

The **Data Distribution Modifier** module adjusts the sample quantity or distribution, but these generated samples are unlabeled. Moreover, to provide real-time feedback to IoT edge devices, rapid training of unseen tasks is necessary. However, timely training of unseen tasks on a large number of samples is challenging. Therefore, we need the assistance of a **Data Representation Extractor** module to extract meaningful information from unlabeled data. This accelerates the training of unseen tasks while enhancing training accuracy.

As outlined in Section II-B, STL is an excellent algorithm for extracting information from unlabeled data. Learning the basic information of unlabeled data in an unsupervised manner is an advantage of STL. We implement the STL module through an autoencoder [38]. Specifically, we train the autoencoder using unlabeled samples, and then utilize the encoding part (*i.e.* encoder) of the autoencoder to extract the data representations of the samples. In an autoencoder, the encoder is designed with deeper layers and fewer neurons. This design forces the network to use a compressed representation of the input data with fewer features. Therefore, the encoder is capable of extracting the data representations of the samples.

Algorithm 2: Extract Data Representations via Self-taught Learning (STL)

Input: The unlabeled data $\{x_u^{(1)}, ..., x_u^{(k)}\}$ with each $x_u^{(i)\in R^n}$ generated by GANs, the original labeled data $\{(x_l^{(1)}, y^{(1)}), ..., (x_l^{(m)}, y^{(m)})\}$.

Output: The new training set $\hat{T}$.

1 Solve the optimization problem 12 with unlabeled data $\{x_u^{(i)}\}$ to obtain bases b.

2 Compute data representations for the unseen tasks to be trained over the original labeled data to obtain a new training set $\hat{T} = \{(\hat{a}(x_l^{(i)}), y^{(i)})\}_{i=1}^{m}$ with bases b and Eq.13.

As depicted in Algorithm 2, applying STL in the **Data Representation Extractor** module involves the following two steps:

1) Given the unlabeled data $\{x_u^{(1)}, ..., x_u^{(k)}\}$ with each $x_u^{(i)\in R^n}$ generated by GANs, STL solves the following optimization problem:

$$\begin{aligned} \min_{b,a} \quad & \sum\nolimits_i \|x_u^{(i)} - \sum\nolimits_j a_j^{(i)} b_j\|_2^2 + \beta\|a^{(i)}\|_1, \\ s.t. \quad & \|b_j\|_2 \leq 1, \forall j \in 1, ..., s. \end{aligned} \tag{12}$$

where $a_j^{(i)}$ is the activation of base b_j for input $x_u^{(i)}$. Sparse coding is applied to the unlabeled data $x_u^{(i)} \in R^s$ to learn a set of bases b.

2) With learned b, for each original labeled samples $x_l^{(i)} \in R^s$ collected from IoT edge devices, $\hat{a}(x_l^{(i)}) \in R^s$ is computed by solving the following optimization problem:

$$\begin{aligned} \hat{a}(x_l^{(i)}) = & \arg\min_{a^{(i)}} \|x_l^{(i)} - \sum\nolimits_j a_j^{(i)} b_j\|_2^2 \\ & + \beta\|a^{(i)}\|_1. \end{aligned} \tag{13}$$

The data representation $\hat{a}(x_l^{(i)})$ of each original labeled sample $x_l^{(i)}$ collected from IoT edge devices is then used for training the unseen task.

The original STL framework employs linear autoencoders. However, in the edge-cloud synergy AI service, the data collected from IoT edge devices often consists of multiple dimensions or channels, rendering linear autoencoders inefficient. Therefore, we innovatively adopt convolutional autoencoders for the implementation of STL, enabling the extraction of feature representations from each channel effectively. Specifically, based on the feature dimensions of the IoT edge device data collected, we design convolutional and deconvolutional structures with varying numbers of layers. For the encoder, we initially expand the number of channels through convolution

to extract data representations. Subsequently, we reduce the number of channels for training unseen tasks. The decoder, on the other hand, employs deconvolution, which operates inversely to the encoder, to assess the effectiveness of the data representations generated by the encoder.

With the role of the **Data Representation Extractor** module based on STL, we obtain a new training dataset $\hat{T}$. This dataset retains the effective information of the original dataset while reducing the dimensionality of the training data. As a result, training convergence is accelerated, and due to the removal of irrelevant information, training accuracy is improved.

D. Unseen Task Trainer

Under the Constraint 4, we cannot utilize the historical tasks. We train the unseen task using the new training dataset $\hat{T}$ which contains extracted data representations and the original labels of the samples collect at the IoT edge devices, ensuring convergence in the training of the unseen task and the expected training loss converges to the minimum value F^* with a ρ precision with the minimum training time T. In what follows, we analyze how our design enables the convergence of training for unseen tasks and overcomes the challenges posed by small sample and heterogeneous sample issues.

To theoretically prove the validity of our design, we first introduce some hypotheses and related conclusions from [39].

Stochastic Update Rule. Without loss of generality, the parameters w are updated by solving the following subproblem at iteration t:

$$w_t = \arg\min_{w\in\Omega}[\phi_{I_t}(w) + \frac{\gamma_t}{2}\|w - w_{t-1}\|_2^2] \tag{14}$$

It consists of two components: the first part minimizes the loss function on mini-batch I_t. The second component is to avoid overfitting.

Assumption of Related Conclusions. The introduced theorem needs the following assumption:

Assumption 1. *We assume that for all t:*

$$\begin{aligned}&\mathbf{E}_{I_t}[D_\phi(w_t; w_{t-1})] \leq \\ &\mathbf{E}_{I_t}[D_{\phi_{I_t}}(w_t; w_{t-1}) + \frac{\gamma_t}{2}\|w_t - w'_{t-1}\|_2^2].\end{aligned} \tag{15}$$

Related Theorem. We introduce the related theorem we utilize:

Lemma 1. *Consider the stochastic update rule 14. Assume that ϕ_i is λ-strongly convex for all i:*

$$\phi_i(w) - \phi_i(w') - \nabla\phi_i(w')^\top(w - w') \geq \frac{\lambda}{2}\|w - w'\|_2^2. \tag{16}$$

Under Assumption 1 and when choosing the update parameter

$$\gamma_t = \gamma + \lambda(t-1), \tag{17}$$

we have for all $w_* \in \Omega$:

$$\sum_{t=1}^{T}\mathbf{E}[\phi(w_t) - \phi(w_*)] \leq \frac{\gamma}{2}\|w_* - w_0\|_2^2 + \frac{A_2}{b}\sum_{t=1}^{T}\frac{1}{\gamma_t}, \tag{18}$$

where $A^2 = \sup_{w\in\Omega} n^{-1}\sum_{i=1}^{n}\|\nabla\phi_i(w) - \nabla\phi(w)\|_2^2$.

Choosing

$$\gamma = \sqrt{\frac{2T}{b}}\frac{A}{\|w_* - w_0\|_2} \tag{19}$$

yields the following aggregate regret bound:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbf{E}[\phi(w_t) - \phi(w_*)] \leq \frac{\sqrt{2}A}{\sqrt{Tb}}\|w_* - w_0\|_2. \tag{20}$$

It is worth noting that, as the related theorem considers centralized training, it does not account for the challenges introduced by the diversity of IoT edge devices, such as small sample or heterogeneous sample issues. Therefore, we have made adjustments to the theory under small sample or heterogeneous sample conditions and present the following theorem:

Theorem 1. *The small sample problem prevents the convergence of training for unseen tasks, while the heterogeneous sample problem leads the model into local bias. Our design addresses both of these issues, setting the regret bounds for training unseen tasks as $\frac{\sqrt{2}A^{recons}}{\sqrt{Tb}}\|w_* - w_0\|_2$.*

Proof. Firstly, we prove that the small sample problem prevents the convergence of training for unseen tasks. The aggregate regret bound 20 shows that mini-batch size b is related to training convergence. Small sample problem means that mini-batch size b is not sufficient, *i.e.*, $\exists b, b \to 0$, which leads to the aggregate regret bound to ∞, and thus training unseen tasks cannot converge.

Subsequently, we argue that the heterogeneous sample problem leads to local bias in the training of unseen tasks. Reviewing Assumption 1, it is observed that since w_t depends on I_t, Assumption 1 requires some $\gamma_t > 0$ to satisfy the condition. However, in the heterogeneous environment introduced by IoT edge devices, I_t varies with the changes in devices, leading to various w_t. Therefore, given that γ is chosen unchangeable or does not adapt well to changes in heterogeneous samples, some parameters struggle to address the issue of overfitting effectively, making w bias.

Finally, we prove that our design ensures the convergence of training for unseen tasks. As we consider heterogeneous samples, we modify Eq.20 to make it associated with the samples X:

$$\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{E}[\phi(w_t; X_i) - \phi(w_*; X_i)] \tag{21}$$

After GANs modify sample quantity and sample distribution, we get reconstructed samples

$$X^{recons} : \{X_1, ..., X_N, G(z^{(1)}), ..., G(z^{(n)})\}, \tag{22}$$

where $\{X_1, ..., X_N\}$ is the original samples, $\{G(z^{(1)}), ..., G(z^{(n)}\}$ is the reconstructed part, with reconstruction quantity n. STL extracts data representations as

$$a(\cdot) : X \to \sum_j a_j^{(i)} b_j. \tag{23}$$

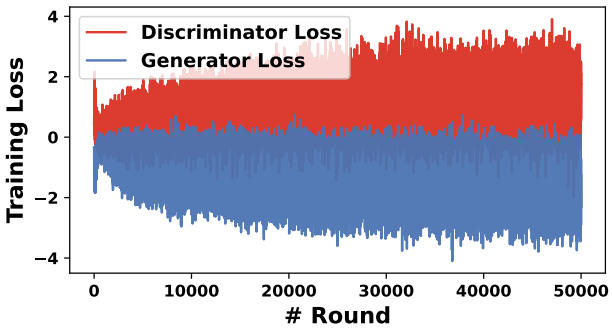

Fig. 2. Training loss of GANs in lifelong learning.

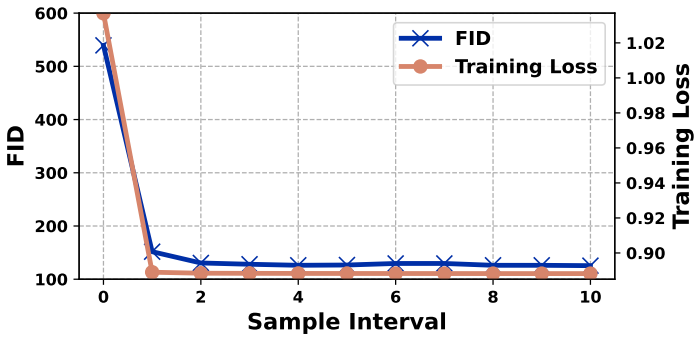

Fig. 3. FID score of GANs (Blue Line) and training loss of STL (Red Line) in 10 sampling intervals. The sampling interval for the FID score of GANs is 5000 rounds. The sampling interval for the training loss of STL is 10 rounds.

Then we have

$$\begin{aligned}&\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{E}[\phi(w_t;X_i)-\phi(w_*;X_i)]\\&=\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{E}[\phi(w_t;a(X_i^{recons}))-\\&\phi(w_*;a(X_i^{recons}))]\\&\leq\frac{\sqrt{2}A^{recons}}{\sqrt{Tb}}\|w_*-w_0\|_2,\end{aligned}\tag{24}$$

where $A^2 = \sup_{w\in\Omega} n^{-1}\sum_{i=1}^{n}\|\nabla\phi_i(w;a(X_i^{recons})) - \nabla\phi(w;a(X_i^{recons}))\|_2^2$. Generally, $A^2 = \sup_{w\in\Omega} n^{-1}\sum_{i=1}^{n}\|\nabla\phi_i(w;a) - \nabla\phi(w;a)\|_2^2$. We prove our design induces training unseen tasks to the regret bound of $\frac{\sqrt{2}A^{recons}}{\sqrt{Tb}}\|w_*-w_0\|_2$. □

By modifying the conditions of small-sample or heterogeneous-sample settings and imposing constraints that historical tasks cannot utilize (*i.e.*, $\exists n, \delta^{[n]} > \epsilon^{[n]}$), we strengthen the convergence boundary towards $\frac{\sqrt{2}A^{recons}}{\sqrt{Tb}}\|w_*-w_0\|_2$. This demonstrates that our design enables convergence for training unseen tasks in the lifelong learning process of edge-cloud synergy AI services based on AIoT, achieving the goal of the expected training loss converges to the minimum value F^* with a ρ precision with the minimum training time T (Constraint 3).

E. Problem P1 Solution Analysis

For Problem **P1**, the small sample issue leads to a failure in convergence, making it impossible to calculate the minimum convergence time. The heterogeneous sample problem results in a violation of Constraint 3, and thus the obtained convergence time does not satisfy the constraint requirements. Our design, while satisfying Constraints 3-6, ensures the optimal convergence time, successfully addressing Problem **P1**.

The **Data Distribution Modifier** module adjusts the sample quantity and distribution. The **Data Representation Extractor** module extracts data representations of the samples and constructs a new training dataset. The **Unseen Task Trainer** module rapidly trains unseen tasks using the new training dataset. These three modules work sequentially, successfully addressing unseen tasks during the lifelong learning process of edge-cloud synergy AI services based on AIoT.

VI. Implementation

To better assess the effectiveness of Seafarer in real-world production environments, we employ an actual and popular edge-cloud synergy AI service KubeEdge-Sedna [9] as our testing platform. KubeEdge-Sedna is an edge-cloud synergy AI project incubated in KubeEdge SIG AI. Benefiting from the edge-cloud synergy capabilities provided by KubeEdge [40], KubeEdge-Sedna can implement across edge-cloud collaborative training and collaborative inference capabilities (*e.g.*, joint inference and lifelong learning), bringing the benefits of reducing costs, improving model performance, and protecting data privacy.

As Seafarer is designed for general-purpose lifelong learning edge-cloud synergy AI services, it can be easily adapted to KubeEdge-Sedna. Seafarer has been implemented in the `unseen_task_processing` module of KubeEdge-Sedna [1] with Pytorch [41]. Specifically, the `task_compare` method original in the **Metaknowledge Base** module compares the metaknowledge of the current task with that of historical tasks. Existing similarity metrics include `cosine_similarity`, `Euclidean_distance`, and *etc*. For the unseen tasks detected by KubeEdge-Sedna, KubeEdge-Sedna sends the collected set of unseen task samples to the **Data Distribution Modifier** module. The `train_GANs` method in the **Data Distribution Modifier** module first checks whether the set of unseen task samples has been trained. If it has, it directly uses the pretrained GAN model; otherwise, it trains the GAN model with the collected set of unseen task samples. Then, the `modify_data` method in the **Data Distribution Modifier** module adjusts the sample distribution and sends the modified samples to the **Data Representation Extractor** module. Similarly, the `train_STL` method in the **Data Representation Extractor** module first checks if the corresponding autoencoder model exists. If not, it trains the autoencoder with the modified samples. Then, the

[1] https://github.com/kubeedge/ianvs/blob/main/examples/cityscapes/lifelong_learning_bench/unseen_task_processing-GANwithSelfTaughtLearning/README.md

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3396282

Fig. 4. Random samples generated by trained GANs.

Fig. 5. Three features of five images extracted by STL.

`extract_representations` method extracts data representations, and the `encoder_data` method constructs a new training dataset. Finally, the `train_unseen_task` method in the **Unseen Task Trainer** module uses the new training dataset to train unseen tasks and returns the results to the **Metaknowledge Base** and **Model Repository**. Then Control is handed back to KubeEdge-Sedna, continuing with the lifelong learning service.

VII. Case Study

A. Setup

We illustrate the effectiveness of our proposed framework Seafarer through a case study. The case study involves the Cityscapes Dataset [22], with the training of the `DeepLabV3` model [42] chosen as the unseen task.

Platform. We prototype our design Seafarer and conduct the simulation experiments on a server equipped with four NVIDIA A100 Tensor Core (40G) GPU cards, an Intel(R) Xeon(R) Gold 6240C CPU (@ 2.60GHz), and 256GB of memory. The cloud server and IoT edge devices are simulated using threads.

Dataset. The Cityscapes Dataset is one of the most authoritative and professional image semantic segmentation benchmark datasets in the field of autonomous driving. It focuses on the understanding of urban road environments in real scenes, making the task more challenging and closely aligned with popular demands such as autonomous driving. Specifically, we choose 5,000 left 8-bit images with dimensions of $2048 * 1024 * 3$ from the dataset.

Model. The `DeepLabV3` model is a semantic segmentation algorithm that discusses the use of dilated convolutions. This allows for a larger receptive field within the framework of cascaded modules and spatial pyramid pooling, enabling the acquisition of multi-scale information. We simulate training the `DeepLabV3` model as an unseen task in the lifelong learning process of edge-cloud synergy AI services.

Metrics. We observe the training progress of the GANs using training loss and assess its convergence using Fréchet Inception Distance (FID) [43]. FID quantifies the overall semantic realism of synthesized data. For STL training convergence, we monitor the training loss. To evaluate the effectiveness of training unseen tasks, we examine both training and validation losses.

Small Sample Environment Construction and Baselines. To validate the effectiveness of our design, we consider three experimental setups:

1) Full Data Training: we utilize the full set of 5,000 images to train the unseen task as a baseline.
2) Small Sample Training: we randomly select 100 images from a pool of 5,000 as training data for the unseen task to simulate the small sample environment.
3) Small Sample Representation Training: we randomly select 100 images from a pool of 5,000 as training data. Before training unseen tasks, we preprocess the small sample using our design Seafarer, *i.e.*, the unseen task is trained over data representation set.

B. Overall Performance

We first present the results of training GANs in the lifelong learning context, followed by showcasing the training outcomes of the STL module. Finally, we demonstrate the overall performance of our design with training loss and validation loss comparison.

The Results of Training GANs in the Lifelong Learning Context. We initially train GANs using small sample data (*i.e.*, 100 images). We conduct a total of 50,000 rounds of training, and the training losses for the generator G and discriminator D are illustrated in Fig. 2. Due to the specially designed network architecture, the generator G does not face the issue of gradient vanishing [44] and receives informative feedback in the adversarial game with the discriminator D. The GAN pair plays games with each other until the end of training with maximum training loss not exceeding 4. Fig. 3 (Blue Line) depicts the quality of images generated by the generator G, demonstrating that the training converges effectively, with the minimum FID score of 125.50. Fig. 4 displays randomly generated images that bear similarity to real images.

The Results of Training STL. We then proceed to train STL using 3,000 images generated by GANs, as depicted in Fig. 3 (Red Line). STL demonstrates convergence under the generated data from GANs with minimum training loss of 0.88. Fig. 5 illustrates the features extracted by the trained STL. The first row of Fig. 5 represents the images generated by GANs, *i.e.*, the input to STL. The second, third, and fourth rows respectively represent the data representations extracted by STL. These data representations capture the essential features of the original data, simplifying the training of unseen tasks.

The Unseen Task Training. Fig. 6 illustrates the training loss for unseen tasks under three experimental settings. It

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3396282

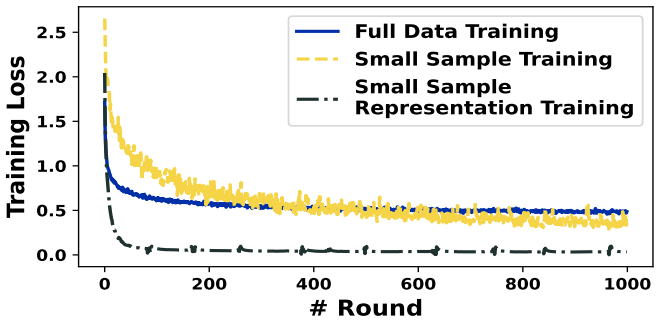

Fig. 6. Training loss of the unseen task in lifelong learning.

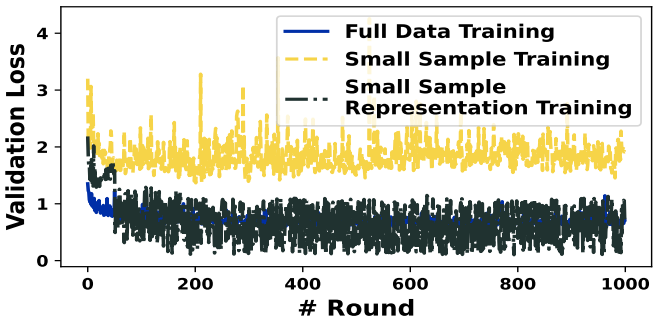

Fig. 7. Validation loss of the unseen task in lifelong learning.

can be observed that training unseen tasks using under small sample scenarios not only results in slow loss reduction but also exhibits unstable oscillations. The training loss variance in the last 500 iterations is 0.08. However, after training with data representations, the loss not only decreases rapidly and stabilizes, but also achieves better results than training with the full data. Specifically, our design converges after 20 iterations, reducing the number of iterations by 90% compared to the process of using the full data for training. Moreover, our design attains a lower training loss (under the same loss function), reducing the training loss by 80% compared to the process of training with the full data. Fig. 7 presents the validation loss for unseen tasks under three experimental settings. The small sample training results in an unstable and consistently non-decreasing validation loss. In comparison to training with the full data, our design initially experiences higher validation loss due to the constraints imposed by small sample training. However, as training iterations progress, the validation loss begins to decrease around 100 iterations. Although the validation loss remains less stable compared to training with the full data, the stability of the validation loss improves in contrast to the small sample training scenario, with a reduction in variance by approximately 33%.

Analysis on Training Time. We train GANs for 50,000 rounds, as shown in Fig. 2, which takes approximately 9 hours. However, as shown in Fig. 3 (Blue Line), the training of GANs converges around 10,000 epochs. Therefore, in practical deployment, training can be reduced to one-fifth of the time, around 1.8 hours. We train the STL model for 100 epochs, as shown in Fig. 3 (Red Line), which takes approximately 7.2 hours. However, as shown in Fig. 3 (Red Line), the training converges around the 15 epoch. Therefore, in practical deployment, training time can be reduced to approximately 1 hour. Considering the large data dimensionality of $2048 * 1024 * 3$ we deal with, the time is acceptable. We reduce the training epochs for unseen tasks through certain preprocessing overheads.

Analysis on Lightweight Model Parameters. We measure the model size using `torchsummary`, as depicted in Table II. The parameter size of the GAN is 143.30MB, with the generator accounting for 111.30MB and the discriminator for 32.00MB. For the STL model, the parameter size is 0.0034MB, with the encoder and decoder each occupying 0.0017MB. Compared to LLMs, our model has a smaller parameter count, making it lightweight.

TABLE II
THE SIZES OF GAN AND STL MODEL PARAMETERS.

Item	GAN		STL	
	Generator	Discriminator	Encoder	Decoder
#Params	111.30MB	32.00MB	0.0017MB	0.0017MB

VIII. DISCUSSION AND FUTURE WORK

Looking ahead, several avenues for future work emerge. First, exploring variations and enhancements to GANs could further improve the quality and diversity of synthetic data, contributing to more effective lifelong learning. Second, investigating the integration of our proposed design into real-world IoT edge-cloud synergy AI deployments would provide valuable insights into its practical feasibility and performance under diverse scenarios. Additionally, addressing the scalability and efficiency of our design in larger and more complex edge-cloud ecosystems is a critical consideration for future research. Evaluating the design across various datasets and application domains will help assess its generalizability and potential limitations. Finally, our modules are currently executed in a serial manner. While ensuring correctness, parallel execution of modules can further enhance the efficiency of training unseen tasks. Our proposal for parallelism is categorized into intra-unseen-task processing and inter-unseen-task processing. Intra-unseen-task processing involves the **Data Distribution Modifier** module, which generates data for each batch, and the **Data Representation Extractor** module, which learns its fundamental representation, forming a pipeline. Inter-unseen-task processing entails initiating a dedicated processing pipeline on the cloud server for each unseen task, allowing different unseen tasks to be processed concurrently.

IX. CONCLUSION

In this paper, our proposed design Seafarer leveraging Generative Adversarial Networks (GANs) and Self-taught Learning (STL) presents a robust solution to the challenges associated with unseen tasks in the lifelong process of edge-cloud synergy AI services based on AIoT. By addressing the

biases inherent in fine-tuning large models on heterogeneous data, our design ensures that models remain unbiased, even when faced with diverse edge-collected datasets. Moreover, our design significantly reduces the demand for extensive training data, providing a scalable solution for the lifelong learning scenarios. The integration of GANs allows for the generation of synthetic data to augment small or heterogeneous samples, overcoming limitations associated with data scarcity and distributional shifts.

In conclusion, our work lays a foundation for advancing the capabilities of edge-cloud synergy AI services based on AIoT in handling unseen tasks and presents an exciting direction for future research to bridge the gap between theoretical advancements and practical deployment scenarios.

ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Program of China (2023YFE0205700), National Natural Science Foundation of China (62341410, 62302348), National Natural Science Foundation of China (62372333), Science and Technology Development Fund, Macao S.A.R (FDCT) project (0078/2023/AMJ), and General Program of Hubei Provincial Natural Science Foundation of China (2023AFB831).

REFERENCES

[1] M. Abdel-Basset, G. Manogaran, A. Gamal, and V. Chang, "A novel intelligent medical decision support model based on soft computing and iot," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4160–4170, 2019.

[2] F. Cirillo, D. Gómez, L. Diez, I. E. Maestro, T. B. J. Gilbert, and R. Akhavan, "Smart city iot services creation through large-scale collaboration," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5267–5275, 2020.

[3] A. D. Boursianis, M. S. Papadopoulou, P. Diamantoulakis, A. Liopa-Tsakalidi, P. Barouchas, G. Salahas, G. Karagiannidis, S. Wan, and S. K. Goudos, "Internet of things (iot) and agricultural unmanned aerial vehicles (uavs) in smart farming: A comprehensive review," *Internet of Things*, vol. 18, p. 100187, 2022.

[4] M. Abrar, U. Ajmal, Z. M. Almohaimeed, X. Gui, R. Akram, and R. Masroor, "Energy efficient uav-enabled mobile edge computing for iot devices: A review," *IEEE Access*, vol. 9, pp. 127 779–127 798, 2021.

[5] X. Chen, M. Li, H. Zhong, Y. Ma, and C.-H. Hsu, "Dnnoff: offloading dnn-based intelligent iot applications in mobile edge computing," *IEEE transactions on industrial informatics*, vol. 18, no. 4, pp. 2820–2829, 2021.

[6] S. Tang, L. Chen, K. He, J. Xia, L. Fan, and A. Nallanathan, "Computational intelligence and deep learning for next-generation edge-enabled industrial iot," *IEEE Transactions on Network Science and Engineering*, 2022.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[8] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1423–1431.

[9] kubeedge, "sedna," 2021. [Online]. Available: https://github.com/kubeedge/sedna

[10] S. Wan, Z. Gu, and Q. Ni, "Cognitive computing and wireless communications on the edge for healthcare service robots," *Computer Communications*, vol. 149, pp. 99–106, 2020.

[11] F. Khan, M. A. B. Siddiqui, A. U. Rehman, J. Khan, M. T. S. A. Asad, and A. Asad, "Iot based power monitoring system for smart grid applications," in *2020 international conference on engineering and emerging technologies (ICEET)*. IEEE, 2020, pp. 1–5.

[12] K. Gulati, R. S. K. Boddu, D. Kapila, S. L. Bangare, N. Chandnani, and G. Saravanan, "A review paper on wireless sensor network techniques in internet of things (iot)," *Materials Today: Proceedings*, vol. 51, pp. 161–165, 2022.

[13] Z. Zheng, P. Luo, Y. Li, S. Luo, J. Jian, and Z. Huang, "Towards lifelong thermal comfort prediction with kubeedge-sedna: Online multi-task learning with metaknowledge base," in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 263–276.

[14] Z. Zheng, Y. Li, H. Song, L. Wang, and F. Xia, "Towards edge-cloud collaborative machine learning: A quality-aware task partition framework," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 3705–3714.

[15] W. Li, Z. Wang, X. Yang, C. Dong, P. Tian, T. Qin, J. Huo, Y. Shi, L. Wang, Y. Gao *et al.*, "Libfewshot: A comprehensive library for few-shot learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[16] Y. Tan, Y. Liu, G. Long, J. Jiang, Q. Lu, and C. Zhang, "Federated learning on non-iid graphs via structural knowledge sharing," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 8, 2023, pp. 9953–9961.

[17] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2998–3009.

[18] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[19] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[20] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[21] F. Gilardi, M. Alizadeh, and M. Kubli, "Chatgpt outperforms crowd-workers for text-annotation tasks," *arXiv preprint arXiv:2303.15056*, 2023.

[22] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] H. Poon and P. Domingos, "Joint inference in information extraction," in *AAAI*, vol. 7, 2007, pp. 913–918.

[24] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 374–382.

[25] Z. Zheng, Y. Dai, and D. Wang, "Duet: Towards a portable thermal comfort model," in *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2019, pp. 51–60.

[26] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, pp. 103–134, 2000.

[27] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[28] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 759–766.

[29] M. Liu, Y. Wei, X. Wu, W. Zuo, and L. Zhang, "Survey on leveraging pre-trained generative adversarial networks for image editing and restoration," *Science China Information Sciences*, vol. 66, no. 5, pp. 1–28, 2023.

[30] M. La Salvia, E. Torti, R. Leon, H. Fabelo, S. Ortega, B. Martinez-Vega, G. M. Callico, and F. Leporati, "Deep convolutional generative adversarial networks to enhance artificial intelligence in healthcare: A skin cancer application," *Sensors*, vol. 22, no. 16, p. 6145, 2022.

[31] J. P. Killgore, T. J. Kolibaba, B. W. Caplins, C. I. Higgins, and J. D. Rezac, "A data-driven approach to complex voxel predictions in grayscale digital light processing additive manufacturing using u-nets and generative adversarial networks," *Small*, vol. 19, no. 50, p. 2301987, 2023.

[32] B. Liu, Y. Zhu, K. Song, and A. Elgammal, "Towards faster and stabilized gan training for high-fidelity few-shot image synthesis," in *International Conference on Learning Representations*, 2020.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[34] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep residual learning in spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 056–21 069, 2021.
[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.
[36] H. Talebi and P. Milanfar, "Learning to resize images for computer vision tasks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 497–506.
[37] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
[38] D. Holden, J. Saito, T. Komura, and T. Joyce, "Learning motion manifolds with convolutional autoencoders," in *SIGGRAPH Asia 2015 technical briefs*, 2015, pp. 1–4.
[39] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 661–670.
[40] kubeedge, "kubeedge," 2018. [Online]. Available: https://github.com/kubeedge/kubeedge
[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
[42] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.
[43] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.
[44] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

Tianyu Tu received the B.S. degree in Software Engineering from Wuhan University in 2022. He is currently working towards the M.S. degree in Computer Science and Technology with the Computer School, Wuhan University. His research interests include federated unsupervised learning, game theory, and AIoT.

Zhili He received the B.S. degree in Automation Science from Chongqing University of Technology in 2006, and received the M.S. degree from Chongqing University in 2021. He is currently pursuing his Ph.D. in Computer Science at Wuhan University. His research interests include Industrial Big Data and IoT.

Zhigao Zheng received his PhD degree in computer science from Huazhong University of Science and Technology. He is currently with the School of Computer Science, Wuhan University, China. His research interests focus on cloud computing, big data processing, and AI systems. He is the editorial board member of some high-quality journals, such as the IEEE Transactions on Consumer Electronics, Mobile Networks and Applications.

Zimu Zheng is currently a head research engineer at Huawei Cloud. He received his B.Eng. degree in South China University of Technology and Ph.D degree in the Hong Kong Polytechnic University. Zimu has received several awards for outstanding technical contributions in Huawei. He also received the Best Paper Award of ACM e-Energy and the Best Paper Award of ACM BuildSys in 2018. His research interest lies in edge intelligence, benchmarking, multi-task learning, and AIoT.

Jiawei Jiang received his PhD degree in computer science from Peking University, China, in 2018. He is currently a professor with the School of Computer Science, Wuhan University, China. His research interests include database, big data management and analytics, and machine learning systems.

Yili Gong received her BS degree in Computer Science from Wuhan University in 1998, and her Ph.D. degree in Computer Architecture from the Institute of Computing, Chinese Academy of Sciences in 2006. She is currently an Associate Professor in the School of Computer Science at Wuhan University. Her interests include intelligent operations and maintenance in HPC environments, distributed file systems and blockchain systems.

Chuang Hu received his BS and MS degrees from Wuhan University in 2013 and 2016. He received his Ph.D. degree from the Hong Kong Polytechnic University in 2019. He is currently an Associate Professor in the School of Computer Science at Wuhan University. His research interests include edge learning, federated learning/analytics, and distributed computing.

Dazhao Cheng (Senior Member, IEEE) received his BS and MS degrees in Electrical Engineering from the Hefei University of Technology in 2006 and the University of Science and Technology of China in 2009. He received his Ph.D. from the University of Colorado at Colorado Springs in 2016. He was an AP at the University of North Carolina at Charlotte in 2016-2020. He is currently a professor in the School of Computer Science at Wuhan University. His research interests include big data and cloud computing.