Temporal Data Pooling With Meta-Initialization for Individual Short-Term Load Forecasting

Eunju Yang, *Student Member, IEEE*, and Chan-Hyun Youn, *Senior Member, IEEE*

***Abstract*—The growth of advanced metering infrastructure (AMI) deployment has enabled intelligent power control and management at the customer level; Highly accurate individual short-term load forecasting is crucial for this precise control per customer. Recently, deep learning has been widely adopted to improve forecasting accuracy. However, training an individual deep network (local training) has overfitting issues due to data paucity per customer. Thus, a pooling scheme has been introduced to augment a training dataset by batching several customers' data. Nevertheless, there is room for existing pooling approaches to further improve accuracy by considering distribution heterogeneity within a customer dataset. In addition, their static pool assignment only with a customer's training dataset may cause accuracy degradation under concept drift in serving time. To overcome these, we propose a Temporal Data Pooling (TDP) that constructs data pools at the data sample level with a novel distribution inference method and theoretical analysis. It allows the most probable forecasting model to serve predictions while resolving data shortage issues in local training. The TDP outperforms the other six competing methods for point and probabilistic forecasting; it shows robust accuracy under concept drift. Moreover, it demonstrates superior accuracy for unseen customers without additional training, proving its scalability.**

***Index Terms*—Temporal data pooling, pooling, load profiling, individual short-term load forecasting, concept drift, meta-initialization.**

I. Introduction

FOR A highly accurate smart grid service, precise short-term load forecasting that predicts load values in minutes or days is crucial [1], [2], [3], [4]. In particular, 'individual' short-term load forecasting (ISTLF), conducted at the customer level, is extremely demanding for customer-specific services, such as peak load shaving and demand-side response [5], [6]. The ISTLF was made possible by the massive deployment of Advanced Metering Infrastructures (AMIs). An AMI meter, installed at each building, periodically measures the energy consumption of the building and transmits it to the Cloud environment managed by smart grid operators, where the data is processed for various energy services.

However, unlike aggregated load forecasting, which is relatively accurate thanks to its regular pattern [7], [8], [9], the ISTLF is more challenging because an individual profile contains non-trivial uncertainty and has volatile properties [10]. Therefore, it is required to study sophisticated technologies; Deep learning has shed light on the precise ISTLF for its ability to learn non-linear features. Therefore, deep networks have been actively explored as the solution to the ISTLF [11], [12].

When applying the deep network to ISTLF for multiple customers, how constructing a forecasting network is an additional challenge [13]. Distribution of individual loads varies by the customer's lifestyle or surroundings, which implies the ISTLF for different customers might be statistically disparate tasks [12]. Thus, how to construct a dataset affects the accuracy of deep network-based ISTLF, for its training follows empirical risk minimization. As the most straightforward way, it has been taken for granted to train a single network for a customer, known as the *one-to-one* method (or *local*) [5], [14].

However, this *local* framework may suffer from an overfitting problem due to the insufficient dataset for each customer [15]. In addition, its scalability is low, requiring the same number of networks as the customer. As a solution to this, training a single network for all customers (so-called, *one-to-all*, or *global*) has recently been spotlighted [16], but with the *global* framework, the practitioner should increase the model complexity to incorporate all conflicted task distributions, increasing model size.

Thus, as the moderation between *local* and *global*, a novel pooling scheme was proposed [17], [18], [19], [20], [21], which trains a model for a set of similar customers by batching training datasets of each group. By sharing one model with similar customers, the pooling-based approaches solve the data scarcity problem and improve resource efficiency with high serving throughput [13]. However, since the pooling schemes consider deviation between customers only, there is room to further improve forecasting accuracy through distribution matching at a sample level. They assign the whole data samples from a customer into one chamber, which ignores distribution deviation among time-series samples within a customer dataset [14]. Moreover, since they assign only a single network to a customer based on its training dataset, they are vulnerable to concept drift, where the distribution is changed at the serving time [22], [23].

Manuscript received 13 September 2021; revised 17 February 2022, 16 July 2022, and 13 November 2022; accepted 26 November 2022. Date of publication 1 December 2022; date of current version 21 June 2023. This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government (MSIT, Core Technology Development for Intelligently Searching and Utilizing Big Data Based on DataMap) under Grant 2020-0-00077. The work of Eunju Yang was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (NRF-2019-Global Ph.D. Fellowship Program) under Grant 2019H1A2A1074414. Paper no. TSG-01480-2021. *(Corresponding author: Chan-Hyun Youn.)*

The authors are with the Department of Electrical and Electronic Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea (e-mail: yejyang@kaist.ac.kr; chyoun@kaist.ac.kr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSG.2022.3225805.

Digital Object Identifier 10.1109/TSG.2022.3225805

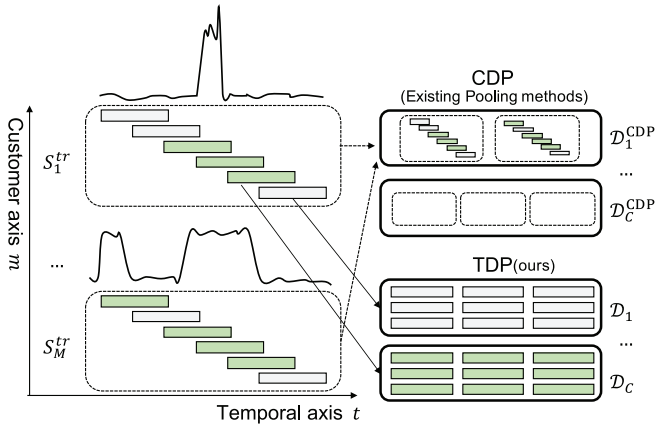


Fig. 1. Conceptual figure to compare customer data pooling (**CDP**), which is existing pooling method to batch across customer dataset, and the **TDP** (proposed). Each data pool is used in training a forecasting model. For the CDP, forecasting model for a customer m is assigned according to where the customer's dataset is allocated in the pool. On the other hand, the TDP framework dynamically allocate the best forecasting model from input sequential vector.

To remedy this, we propose a **T**emporal **D**ata **P**ooling (TDP) framework that adopts the time-series sample-level clustering to pooling. It analyzes every sliding window samples in all customer datasets to reduce sample deviation within a pool rather than simply lumping similar customers' datasets. For clarification, Fig. 1 compares the existing pooling schemes with TDP; existing pooling approaches only reconstruct data pools along the customer axis. Meanwhile, TDP redefines forecasting tasks by analyzing every data sample over time in all customers' datasets, i.e., taking both customer and temporal axes into account. By doing so, TDP can mitigate distribution deviation within a pool while also taking advantage of pooling.

Furthermore, we present a theoretical rationale for the TDP by comparing it with the existing pooling methods with the statistical learning framework. It supports our motivation to combine time series sample clustering and dataset batching across multiple customers. To incarnate the TDP framework, we present two modules. First, we propose a variational inference-based recurrent deep embedding (VaRDE) method for precise time-series sample clustering. Second, meta-initialization [24] is adopted to compensate for possible pool size deviation problems. This end-to-end TDP framework contemplates both non-stationary property in an individual load and the data paucity issue of local ISTLF based on the generalization bound. This paper extends our precedent study [13]; we provide further empirical study and theoretical analysis. Here is the summary of our contributions.

- We analyze the limitation of existing pooling methods in terms of generalization bound and revisit the sample-level clustering as a solution for distribution-aware pooling.
- As a part of the TDP framework, we propose a VaRDE network tailored to stochastic time-series sample clustering and present an analysis of meta-initialization.
- We provide an ablation study to validate each component in the end-to-end TDP framework.

II. Related Works

A. Individual Short-Term Load Forecasting

Load forecasting is the basis for optimized decision-making, such as purchasing electricity in the smart grid. In particular, individual load forecasting, computed per customer, can be used to build a smart home energy management system combined with IoT [25] or provide customized services such as transactive systems [26], [27]. Load forecasting is categorized into short-term, medium-term, and long-term according to the time interval from input to the forecasting target. Among the categories, short-term load forecasting predicts values after several minutes to several days. It is challenging to obtain high accuracy for short-term loads by an individual customer due to volatile properties and high uncertainties [17]. Therefore, some studies were conducted mainly on aggregate load forecasting to mitigate the impact of uncertainty [28], [29]. Following the widespread deployment of AMIs, studies based on the characteristics of each smart meter revealed higher accuracy in predicting aggregated load. Zhang et al. [30] used support vector machine models corresponding to each pattern through hierarchical clustering from various attributes of individual customers and predicted the final accumulative load. Quilumba et al. [31] also leveraged smart meter data to improve aggregated load forecasting performance by building neural networks for each group formed based on the similarity of customers' consumption patterns. Stephen et al. [8] found that daily load contains sub-profiles within customer data. Based on that, they attempt to improve the aggregated load forecasting accuracy by predicting the next day's sub-profile with transition probability between daily sub-profiles. However, because they remained focused on aggregate load forecasting, they did not evaluate the forecasting accuracy for individual loads.

Individual short-term load forecasting is more challenging than aggregated load forecasting because it is fought with uncertainty and lacks general patterns. However, deep networks, which can learn non-linear features, have the potential to solve this problem by learning uncertainty features shared among customers [17]. Especially, recurrent neural network (RNN) is good at learning sequential features; Kong et al. [6] proposed individual load forecasting based on a long-term short-term (LSTM) network. The LSTM, which was trained for each customer, performed better than the previous schemes in the ISTLF. Kim and Cho [32] proposed the CNN-LSTM architecture for forecasting energy consumption. Chen et al. [33] proposed a deep residual network-based short-term load forecasting technique. Probabilistic forecasting is necessary due to the individual load profile's stochastic features and application requirements. Wang et al. [34] proposed probabilistic forecasting using pinball loss, and Yang et al. [18] proposed Monte-Carlo (MC) dropout [35] as a Bayesian deep network approximation for probabilistic forecasting.

B. Pooling-Based Individual Short-Term Load Forecasting

The pooling technique resolves the overfitting issue of conventional local ISTLF caused by the lack of data for each target customer. Furthermore, it is easy to implement and

reduces the number of models while mitigating the issue. Shi et al. [17] first proposed the novel pooling concept, which batches two or more customer data sets into one training dataset for a single network. By doing so, pooling-based ISTLF constructs fewer deep learning models than the total number of customers, allowing each network's model complexity to increase for better generalization. However, the authors did not severely care about grouping customers but used random pooling, configuring a pool without considering customer data characteristics. This random pooling may rather degrade accuracy due to distribution collision in one pool [19].

Its downstream studies have been vigorously conducted for sophisticated pooling. They commonly focused on finding similar customers using clustering. Yang et al. [18] applied agglomerative hierarchical clustering to customer profiles, where a customer profile is constructed by averaging every weak load of the customer. Zang et al. [20] used mutual information for pooling. Rather than grouping all customers, they seek customers who have high mutual information with a target customer. Han et al. [19] performed pooling using K-Means clustering. By clustering customers' residential load profiles, they performed normalization on one pool and learned the LSTM network. In addition, several clustering methods of load profiling [36], [37] can be directly applied to pooling, allowing similar customers to make a colony by classifying customers' profiles. As they allocate the entire data collected from one customer into one chamber [38], they can be seen as a sort of customer pooling. Although these sophisticated clustering techniques have improved the efficiency of pooling, they are still confined to batching *inter*-customer datasets, lacking *intra*-customer analysis.

C. Time-Series Clustering in Load Forecasting

One of the main obstacles of ISTLF is the change in the data distribution within the customer, that is, the non-stationary feature. *Local* ISTLF studies have fought with this time series change, from seasonal effect analysis to daily pattern change. Bedi and Toshniwal [39] proposed clustering time-series data to look up similar months before training forecasting networks. Accordingly, each trained model can cope with different characteristics of seasonal effects. Still, their clustering process is constrained to finding similar months only, which can not cover all sequences of inputs at a fine-grained level.

Clustering for daily profiles is also explored to incorporate daily change in forecasting. For example, Teeraratkul et al. [40] proposed a shape-based clustering to find the best load pattern for the next day by translating the historical profiles to the sequence of cluster means. Similarly, Stephen et al. [8] split a customer's load profiles into sub-profiles and constructed groups across multiple customers' sub-profiles to gather more information on possible load profiles. However, they are limited to applying to general deep learning-based ISTLF, for they work with the transition probability model. In addition, studies have explored the daily profile clustering result in deep network-based forecasting. Hsaio [14] proposed time-series sample-level clustering to cope with distribution change within a customer. To this end, multiple networks are trained corresponding to clusters constructed from an individual customer. However, since one customer's dataset is split to form multiple sub-datasets, each network inevitably suffers from a severe data shortage. Above all, this *local* strategy trains more models than the number of customers, reducing scalability. On the other hand, our TDP framework takes temporal changes and batching across customers simultaneously, allowing adaptive responses to distribution changes while resolving data paucity issues.

III. Problem Formulation

A. Notations and Assumptions

In this subsection, we define some notations used for the problem description. Here, we posit a situation where M customers are enrolled in the system, so their training datasets $\boldsymbol{S} \coloneqq \{S_m\}_{m=1}^{M}$ are collected beforehand. From the total training set $\boldsymbol{S}$, we aim to provide individual short-term load forecasting to more than M customers with C ($\leq M$) models $\{h_k\}_{k=1}^{C}$ for resource-efficient serving while mitigating concept drift.

Let x_t^m be the load value of customer m at time t, which follows the load data distribution of customer m at time t, i.e., $x_t^m \sim p_t^m(x)$. Then, a short-term load forecasting model is defined as $h : \mathcal{X} \to \mathcal{Y}$. Here, h takes $\boldsymbol{x_t^m} = [x_{t-L+1}^m, \ldots, x_t^m] \in \mathcal{X} \subset \mathbb{R}^L$ as an input, which is a look-back vector from time step t of customer m with L successive past values; the input sequence vectors $\boldsymbol{x}$ are constructed by the sliding window method [22]. Then, the ISTLF model returns a single value $y_t^m (= \hat{x}_{t+T}^m) \in \mathcal{Y} \subset \mathbb{R}$ after the T time step from the current time t. For the hypothesis finding, we denote loss function as $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. Especially, we pick the squared loss that is $\mathcal{L}_2(y, y') = |y - y'|^2$. For the simplicity of the analysis, we assume the loss function is bounded by one. We assume that the training datasets for M customers are given prior; each customer m's training dataset S_m is given as

$$S_m = \left\{ \left(\boldsymbol{x}_t^m, y_t^m\right) \middle| \left(\boldsymbol{x}_t^m, y_t^m\right) \sim p_t^m(\boldsymbol{x}, y), 0 \leq t \leq T_m^{tr} \right\},$$

where T_m^{tr} denotes the maximum value of the relative time index for customer m's training dataset, corresponding to the number of data samples in S_m.

In order to resolve the overfitting issue caused by data scarcity, pooling techniques construct C training sets by batching training samples in $\boldsymbol{S}$. We denote the k^{th} sub-dataset as a *data pool* $\mathcal{D}_k$. The total data pools $\{\mathcal{D}_k\}_{k=1}^{C}$ satisfy the following conditions:

$$\cup_{k=1}^{C} \mathcal{D}_k = \cup_{m=1}^{M} S_m, \ \mathcal{D}_i \cap \mathcal{D}_j = \varnothing, \ \forall i \neq j.$$

From the data pools, C forecasting models $\{h(\cdot; \mathcal{W}_k)\}_{k=1}^{C}$ are trained, where each model $h(\cdot; \mathcal{W}_c)$ is trained with its corresponding data pool $\mathcal{D}_c$ through empirical risk minimization; $\mathcal{W}_c$ denotes the weight parameter for pool c.

In the pooling-based ISTLF, a data pool represents the training dataset of the corresponding forecasting task. Thus, the way we construct data pools shall determine task distributions. In terms of this, pooling-based ISTLF's performance is determined by how the task distribution is defined and the quality of its data pool, e.g., whether data samples within a pool satisfy i.i.d. conditions. Therefore, we resolve this data

pooling problem by identifying C task distributions that can describe the total training dataset and construct the data pool on the distributions. However, a task distribution of the ISTLF model, i.e., joint distribution between input and output, can not be accessible in serving time. Thereby, building the data pools based on the joint distribution is impractical. Instead, we presume that input distribution can approximate the joint distribution and focus on finding C conditional distributions of input. This assumption is plausible in that this paper focuses on short-term forecasting.

Assumption 1: The task distribution of an individual short-term load forecasting, i.e., joint distribution $p(\boldsymbol{x}_t, y_t)$, can be approximated with $p(\boldsymbol{x}_t)$.

Assumption 2: C distributions exist in the whole training dataset $\boldsymbol{S}$; Let $p(\boldsymbol{x}|c)$ be the c^{th} distribution. For the sake of simplicity, we abbreviate $p(\boldsymbol{x}|c)$ as p_c.

B. Limitation of Existing Pooling-Based ISTLF Methods

1) Limited Concern on the Concept Drift: Most existing pooling-based ISTLF methods [17], [18], [19], [20] commonly batch datasets over the customer axis (Fig. 1). Hence we term them Customer Data Pooling (CDP). For the existing CDP, an i^{th} data pool is constructed with the following batching rule:

$$\mathcal{D}_i^{\text{CDP}} = \bigcup_{m \in \{m | \mathcal{A}^{\text{CDP}}(S_m) = i\}} S_m, \tag{1}$$

where $\mathcal{A}^{\text{CDP}}$ is a clustering algorithm to assign customer m into a specific pool by taking the customer's dataset S_m as an input. The algorithm can be random [17], k-Means [19], or agglomerative hierarchical clustering [18]. The CDP method assigns customer m into a pool c^* based on analyzing the customer's training dataset $c^* := \mathcal{A}^{\text{CDP}}(S_m)$, which implies that it decides the pool of customer m at the training cycle. This suggests that the forecasting error of $h_{c^*}(\boldsymbol{x}_k^m)$ may increase at a serving time $k \geq T_m^{tr}$ due to concept drift, i.e., $p_t^m(\boldsymbol{x}) \neq p_k^m(\boldsymbol{x}), t \leq T_m^{tr}$ [22], [23]. Therefore, we need a workaround to handle distribution change at serving time.

2) Pooling Method for Distribution Matching Is Required: In addition, a pool's distribution discrepancy may raise the CDP's generalization error. Because the CDP studies usually assume that one customer's training dataset has an identical distribution, they allocate a pool for the whole samples of one customer's dataset, where the pool is chosen using a representative profile [18]. However, because of the non-stationary property of individual load, it cannot be guaranteed that all training samples of the same customer will follow the same distribution, but a customer's load profiles can be described with different distributions over time [8]. Therefore, it is more likely that customer m's training set consists of data points from different distributions along the temporal axis.

Assumption 3: A customer m's dataset S_m consists of non-i.i.d. data which are factorized into $\cup_{c=1}^{C} S_{m,c}$, where $S_{m,c}$ is a sub-set of S_m following distribution p_c, i.e., $S_{m,c} = \{(\boldsymbol{x}, y) \,|\boldsymbol{x} \sim p_c, (\boldsymbol{x}, y) \in S_m\}$.

From Assumption 3, we can approximate a customer m's distribution $\hat{p}^m$ as a weighted summation of C distributions,

$$\hat{p}^m = \sum_{c=1}^{C} \frac{|S_{m,c}|}{\sum_{k=1}^{C} |S_{m,k}|} \cdot p_c = \sum_{c=1}^{C} \frac{|S_{m,c}|}{|S_m|} \cdot p_c. \tag{2}$$

Using (1) and (2), we can draw an empirical distribution of the i^{th} CDP pool $\mathcal{D}_i^{\text{CDP}}$'s distribution as

$$\hat{p}_i^{\text{CDP}} = \sum_{c=1}^{C} \frac{\sum_{m \in \text{CDP}_i} |S_{m,c}|}{\sum_{m \in \text{CDP}_i} |S_m|} \cdot p_c = \sum_{c=1}^{C} \bar{\Delta}_{i,c}^{\text{CDP}} \cdot p_c,$$

where $\text{CDP}_i = \{m | \mathcal{A}^{CDP}(S_m) = i\}$, and $\bar{\Delta}_i = [\bar{\Delta}_{i,c}^{\text{CDP}}]_{c=1}^{C}$ is a simplex over C.

With these representations, we can analyze the performance of the short-term load forecasting model trained with the CDP data pools. Let h_p is a minimizer with respect to distribution p, i.e., $h_p = \arg\min_{h \in \mathcal{H}} \mathcal{R}_p(h)$, and $\mathcal{R}_p(h)$ is an expected risk of given hypothesis for given distribution p, i.e., $\mathbb{E}_{(\boldsymbol{x},y)\sim p}[\mathcal{L}(h(\boldsymbol{x}), y)]$. Let $\hat{h}_i^{\text{CDP}}$ as an empirical risk minimizer with respect to distribution $\hat{p}_i^{\text{CDP}}$. Based on Assumption 1 and statistical learning theory [21], [41], for any $\delta > 0$, the generalization error of $\hat{h}_i^{\text{CDP}}$ for input following p_c satisfies the condition below with a probability of at least $1 - \delta$:

$$\begin{aligned} &\mathcal{R}_{p_c}\left(\hat{h}_i^{\text{CDP}}\right) - \mathcal{R}_{p_c}\left(h_{p_c}\right) \\ &= \mathcal{O}\left(\frac{\sqrt{d + \log 1/\delta}}{\sqrt{|\mathcal{D}_i^{\text{CDP}}|}}\right) + \text{disc}_{\mathcal{H}}\left(\hat{p}_i^{\text{CDP}}, p_c\right), \end{aligned} \tag{3}$$

where d denotes the pseudo dimension of the hypothesis class $\mathcal{H}$; $\text{disc}(p, q)$ represents the discrepancy between the given two distributions, i.e., $\text{disc}_{\mathcal{H}}(p, q) = \max_{h \in \mathcal{H}} |\mathcal{R}_p(h) - \mathcal{R}_q(h)|$.

Eq. (3) demonstrates that the CDP reduces the upper bound of its generalization error compared to the *local*'s, for it increases the number of training samples, i.e., $|\mathcal{D}_i^{\text{CDP}}| = \sum_{m \in \text{CDP}_i} |\mathcal{S}_m| \geq |S_m|$. However, if the distribution discrepancy $\text{disc}(p_c, p_i^{\text{CDP}})$ is dominant, it may increase the risk of CDP-based forecasting. For example, assume that a customer M suffers from concept drift after training — the new input $\boldsymbol{x}_t^M$ follows p_C, but $\bar{\Delta}_{\mathcal{A}^{\text{CDP}}(S_M),C}^{\text{CDP}}$ is zero. Then, the discrepancy term $\text{disc}_{\mathcal{H}}(p_C, p_M^{\text{CDP}})$ must be greater than zero. Moreover, even in the case of $\bar{\Delta}_{\mathcal{A}^{\text{CDP}}(S_M),C}^{\text{CDP}} > 0$, it is not guaranteed that the discrepancy term is zero. Therefore, we should consider dynamic pool assignment and distribution discrepancies within a pool to train ISTLF models with low expected risk. To this end, we propose a new pooling framework considering distribution changes over time and a novel clustering method based on a probabilistic model with latent embedding and distribution inference.

IV. Temporal Data Pooling Framework With Meta-Initialization

A. Temporal Data Pooling

As described in the previous section, removing the discrepancy term in Eq. (3) tightens the error bound of the ISTLF. To this end, we construct each training data pool with samples following the same distribution and assign the most likely pool

to each forecasting input on the fly, namely TDP. Specifically, the TDP breaks up all data in $\boldsymbol{S}$ and reconstructs them into new data pools based on each data distribution instead of batching training set $\{S_m\}_{m=1}^M$ across customer index m only. Since it batches along both the customer m and the temporal t axes, we call this *Temporal Data* pooling. Fig. 1 demonstrates how the CDP and the TDP work differently in data pool construction. In concrete, we can define the temporal data pool as follows.

Definition 1 (Temporal Data Pool): From given multiple customers' datasets $\boldsymbol{S} := \cup_{m\in[M]} S_m$, Temporal Data Pool $\mathcal{D}_i^{\mathrm{TDP}}$ ($i \in [C]$) is defined as

$$\mathcal{D}_i^{\mathrm{TDP}} = \left\{ (\boldsymbol{x}, y) \middle| p(\boldsymbol{x}) = p_i \text{ and } (\boldsymbol{x}, y) \in \boldsymbol{S} \right\}. \tag{4}$$

By Definition 1, the c^{th} temporal data pool is represented with $\mathcal{D}_c^{\mathrm{TDP}} = \cup_{m=1}^M S_{m,c}$, and its distribution $\hat{p}_c^{\mathrm{TDP}}$ is p_c. After this pool construction, C forecasting models are trained with each corresponding data pool $\{\mathcal{D}_i^{\mathrm{TDP}}\}_{i=1}^C$. Let $\hat{h}_i^{\mathrm{TDP}}$ denote a minimizer with respect to distribution $\hat{p}_i^{\mathrm{TDP}}$ minimizer of pool i. Then, the TDP framework provides forecasting to input $\boldsymbol{x}$ with the most probable forecasting model $\hat{h}_{c^*}^{\mathrm{TDP}}$, where $c^* = \arg\max_{c\in[C]} p(c|\boldsymbol{x})$. Since the TDP framework allocates a forecasting model only based on the current input distribution, it can react to distribution change. Therefore, if any target input $\boldsymbol{x}_t$ follows distribution among $\{p_k\}_{k=1}^C$, the TDP can return accurate forecasting results with at least probability. Moreover, Definition 1 allows pool samples to follow the same distribution, reducing the discrepancy term. Thus, if Assumption 4 holds, it is guaranteed that the TDP has a lower generalization error than the CDP.

Assumption 4: The size of every data pool for TDP and CDP are the same, i.e., $|\mathcal{D}_c^{\mathrm{TDP}}| = |\mathcal{D}_i^{\mathrm{CDP}}|\ \forall c \in [C], i \in [N]$, where N is the total number of data pools of the CDP.

Lemma 1: According to Assumptions 2 and 4, the TDP provides a customer m with a more accurate ISTLF result than the CDP, for any $\delta > 0$ with the probability of at least $1-\delta$, where its input $\boldsymbol{x}_t^m$ follows any distribution p_c in $\{p_c\}_{c=1}^C$.

Proof of Lemma 1: Let a customer m's input $\boldsymbol{x}_t^m$ at serving time $t > T_m^{tr}$ follows distribution $p_c \in \{p_i\}_{i=1}^C$. Then, the expected error of $\boldsymbol{x}_t^m$ for the CDP forecasting model $\hat{h}_i^{\mathrm{CDP}}$ is given as Eq. (3), where $m \in \mathrm{CDP}_i$. On the one hand, with the probability of at least $1-\delta$, the expected error of the TDP's minimizer $\hat{h}_{c^*}^{\mathrm{TDP}}$ for $\boldsymbol{x}_t^m$ is given as follows:

$$\begin{aligned}
&\mathcal{R}_{p_c}\left(\hat{h}_{c^*}^{\mathrm{TDP}}\right) - \mathcal{R}_{p_c}\left(h_{p_c}\right) \\
&= \mathcal{O}\left(\frac{\sqrt{d+\log 1/\delta}}{\sqrt{|\mathcal{D}_{c^*}^{\mathrm{TDP}}|}}\right) + \mathrm{disc}_{\mathcal{H}}\left(\hat{p}_{c^*}^{\mathrm{TDP}}, p_c\right).
\end{aligned} \tag{5}$$

Since $c^* = c$ by Assumption 2 and $\hat{p}_c^{\mathrm{TDP}} = p_c$, the latter discrepancy term disappears. To compare the generalization accuracy, we compare the upper bound in (3) and (5):

$$\begin{aligned}
&\mathcal{O}\left(\frac{\sqrt{d+\log 1/\delta}}{\sqrt{|\mathcal{D}_i^{\mathrm{CDP}}|}}\right) + \mathrm{disc}_{\mathcal{H}}\left(\hat{p}_i^{\mathrm{CDP}}, p_c\right) \\
&= \mathcal{O}\left(\frac{\sqrt{d+\log 1/\delta}}{\sqrt{|\mathcal{D}_{c^*}^{\mathrm{TDP}}|}}\right) + \mathrm{disc}_{\mathcal{H}}\left(\hat{p}_i^{\mathrm{CDP}}, p_c\right) \text{ by Assumption 4} \\
&= \mathcal{O}\left(\frac{\sqrt{d+\log 1/\delta}}{\sqrt{|\mathcal{D}_{c^*}^{\mathrm{TDP}}|}}\right) + \mathrm{disc}_{\mathcal{H}}\left(\bar{\Delta}_{i,c}^{\mathrm{CDP}} p_c + \sum_{k\neq c} \bar{\Delta}_{i,k}^{\mathrm{CDP}} p_k, p_c\right) \\
&\geq \mathcal{O}\left(\frac{\sqrt{d+\log 1/\delta}}{\sqrt{|\mathcal{D}_{c^*}^{\mathrm{TDP}}|}}\right) + \mathrm{disc}_{\mathcal{H}}(p_c, p_c) \qquad \blacksquare
\end{aligned}$$

Lemma 1 states that the TDP can find a more accurate hypothesis for each temporal data pool from the same hypothesis class in comparison to the CDP. In addition, if Assumption 2 holds for drifted input, the TDP can provide highly accurate ISTLF with $\hat{h}_{c^*}$ even with the concept drift. The subsequent parts describe how to put this theoretical paradigm into practice.

B. Distribution-Inference for Accurate TDP

What makes *Lemma 1* feasible is to find C distributions $\{p_k\}_{k=1}^C$ satisfying *Assumption 2* from the given dataset $\boldsymbol{S}$. Once we have identified C distributions, we can build the temporal data pools. That is to say, we construct the temporal data pool in the following manner:

$$\tilde{\mathcal{D}}_i^{\mathrm{TDP}} = \left\{ (\boldsymbol{x}, y) \middle| \arg\max_{c\in[C]} p(c|\boldsymbol{x}) = i \text{ and } (\boldsymbol{x}, y) \in \boldsymbol{S} \right\} \tag{6}$$

For realization, we must be able to infer $p(c|\boldsymbol{x})$. Even though a direct clustering method can be exploited as a workaround to infer the pool index c from $\boldsymbol{x}$, it is hard to cluster them due to the high dimensionality of load profiles directly [42]. Despite efforts to cluster after dimension reduction, their consecutive action may lose critical information during the reduction step [43]. On the contrary, to get the probabilistic model without losing the essential information, we concurrently find the hidden features and their clustering information via the deep embedding technique. For that purpose, we posit a generative model representing both dimensional reduction and pool information to determine their joint distribution through variational inference.

1) Hierarchical Generative Model: We use a hierarchical Gaussian Mixture Model (GMM) as the generative model of an individual load vector $\boldsymbol{x}$ inspired by Jiang et al. [44]. Assume that a hidden feature $\boldsymbol{z} \in \mathbb{R}^H$ affects the input vector $\boldsymbol{x} \in \mathbb{R}^L$; $\boldsymbol{z}$ is also decided by a pool index $\boldsymbol{c}$. To be concrete, $\boldsymbol{c}$ follows a categorical distribution parameterized by π_c, where π_c is a simplex describing the prior of every data pool. The hidden feature $\boldsymbol{z}$ follows the GMM described by the pool index c. Then, the input data $\boldsymbol{x}$ is generated from $\boldsymbol{z}$ following multi-variate Gaussian distribution. We exploit a deep learning model to infer the distribution parameters to increase the expressiveness of distribution for the load profiles. Let $g(\cdot)$ and $\boldsymbol{\psi}$ denote the generative network and its parameters, respectively. Then the overall generative model is described as

$$\begin{aligned}
p(c) &= Cat(\boldsymbol{\pi}_c), \\
p(\boldsymbol{z}|c) &= \mathcal{N}(\boldsymbol{z}|\mu_c, \Sigma_c), \\
p_{\psi}(\boldsymbol{x}|\boldsymbol{z}) &= \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x\right),
\end{aligned}$$

where $[\boldsymbol{\mu}_x, \log \boldsymbol{\Sigma}_x] = g(\boldsymbol{z}; \boldsymbol{\psi})$; the distribution parameters for $\boldsymbol{z}$ are $\boldsymbol{\mu}_c \in \mathbb{R}^H$, $\boldsymbol{\Sigma}_c = \boldsymbol{\sigma}_c \cdot \mathbb{I}$ and $\boldsymbol{\sigma}_c \in \mathbb{R}^H$, respectively. Fig. 2 contains this hierarchical model. The solid lines correspond

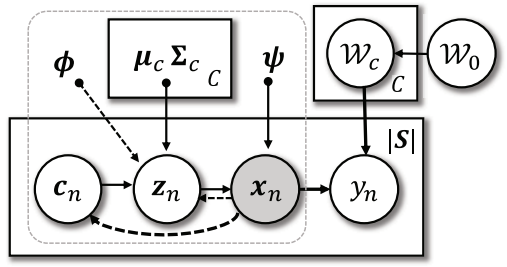


Fig. 2. The probabilistic graphical model of the TDP framework. The solid and dashed lines denote the generative and inference processes, respectively. The arrows (links) demonstrate that a variable in the starting point of the arrow is a condition for the targeting variable. The white and gray circles present unseen and observed random variables, respectively. The gray dashed box represents the graphical model for VaRDE. Each rectangle denotes a data plane with the total number of data presented below.

to this generative relationship. The hierarchical graphical model indicates that the joint distribution is factorized into $p(\boldsymbol{x}, z, c) = p(c)p(z|c)p(\boldsymbol{x}|z)$.

2) Inference Model for $q(z, c|\boldsymbol{x})$: We ultimately find a joint posterior of c and z based on this generative model. However, this posterior $p(z, c|\boldsymbol{x})$ is intractable due to its denominator $p(\boldsymbol{x})$. Thus, instead of directly discovering this true distribution, we find an approximate distribution $q(c, z|\boldsymbol{x})$ through optimization, known as variational inference. To waive this variational inference, we further adopt the mean-field assumption.

Assumption 5 (Mean-Field Assumption): Latent vector z and pool index variable c are conditionally independent, which means the joint conditional distribution is factorized with respective conditional, i.e., $q(z, c|\boldsymbol{x}) = q(z|\boldsymbol{x})q(c|\boldsymbol{x})$.

Assumption 5 makes our variational inference problem easier to solve by turning it into optimization for each variational distribution $q(z|\boldsymbol{x})$ and $q(c|\boldsymbol{x})$. From there, we complete the temporal data pool $\{\tilde{D}_i\}_{i=1}^{C}$ by substituting $p(c|\boldsymbol{x})$ in Definition 1 with the optimal $q(c|\boldsymbol{x})$ taken by the variational inference. To apply optimization to these approximate distributions, we must select the form of each variational posterior. We posit that the variational posterior of z is a Gaussian, as is the generative model of z. The distribution parameters of z are decided by an inference network f parameterized with $\boldsymbol{\phi}$:

$$q(z|\boldsymbol{x}) = \mathcal{N}(z; f(\boldsymbol{x}; \boldsymbol{\phi})), \tag{7}$$

where $[\boldsymbol{\mu}_z, \log \boldsymbol{\Sigma}_z] = f(\boldsymbol{x}; \boldsymbol{\phi})$. Finally, we find its optimum for the variational posterior of c through gradient descent for the variational inference objective.

3) Network Architecture for Distribution Inference of Individual Load Input: Before deriving the optimal $q(c|\boldsymbol{x})$, we propose the structure of both inference and generative network, expressing load vector $\boldsymbol{x}$. Existing variational deep embedding [44] employs simple feed-forward networks for both f and g. But, the fully connected layers are not ideal for individual load data because its clustering does not converge. We conjecture that this is because the input is time-series data acquired through a sliding-window method. A fully-connected network learns features for each neuron without considering the correlation within an input vector. However, in the case of input vectors, the relative values within the vector should be analyzed because not all values within the vectors have absolute positions but are continuously incoming stream data. Thus, hidden embedding z should reflect sequential relationships of the input vector. To this end, we propose a new architecture describing the load data distribution by adopting Gated Recurrent Units (GRUs) [45] to extract hidden features from $\boldsymbol{x}_t$ and recover its information from z. The inference network f takes $\boldsymbol{x}_t$ as input and infers the mean $\boldsymbol{\mu}_z$ and variance $\boldsymbol{\Sigma}_z$ of hidden embedding z_t with the following network:

$$\begin{aligned}
\boldsymbol{h}^{(k)} &= GRU\left(x_k, \boldsymbol{h}^{(k-1)}\right), \\
\boldsymbol{\mu}_z &= ReLU\left(\boldsymbol{W}_\mu h^{(t)} + \boldsymbol{b}_\mu\right), \\
\boldsymbol{\Sigma}_z &= ReLU\left(\boldsymbol{W}_\Sigma h^{(t)} + \boldsymbol{b}_\Sigma\right), \\
z_t &\sim \mathcal{N}\left(z; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z\right),
\end{aligned} \tag{8}$$

where $h^{(k)}$ is a hidden feature of GRU cell with D dimension for input point x_k. $\boldsymbol{W}_\mu$ and $\boldsymbol{W}_\Sigma$ are both matrices with $D \times H$ dimensions for mean and variance, respectively. With the additional bias $\boldsymbol{b}_\mu$ and $\boldsymbol{b}_\Sigma$, the parameters $\boldsymbol{\mu}_z$ and $\boldsymbol{\Sigma}_z$ for the conditional distribution are returned through a fully connected layer with ReLU activation. In the remainder of this study, all parameters of the inference network are shortened with $\boldsymbol{\phi}$.

The generator network g takes z_t as input and returns reconstructed input $\hat{\boldsymbol{x}}_t$, which corresponds to the mean of distribution $p(\boldsymbol{x})$ in the following manner:

$$\begin{aligned}
\hat{\boldsymbol{h}}^{(0)} &= \boldsymbol{W}_z z_t + \boldsymbol{b}_z, \\
\hat{x}_0 &= ReLU\left(\boldsymbol{W}_{out}\hat{\boldsymbol{h}}^{(0)} + \boldsymbol{b}_{out}\right), \\
\hat{\boldsymbol{h}}^{(k)} &= GRU\left(\hat{\boldsymbol{x}}_{k-1}, \hat{\boldsymbol{h}}^{(k-1)}\right), \\
\hat{x}_k &= ReLU\left(\boldsymbol{W}_{out}\hat{\boldsymbol{h}}^{(k)} + \boldsymbol{b}_{out}\right).
\end{aligned} \tag{9}$$

Please note that the generative network is iteratively conducted to get a fully reconstructed vector of $\boldsymbol{x}_t$ for L time steps. Similar to the inference network, to remove abuse of notations, we abridge all parameters in the generative network into $\boldsymbol{\psi}$. Fig. 3 demonstrates the inference and generative network architecture, which we refer to as variational recurrent deep embedding (VaRDE).

4) Training VaRDE to Find the Best Approximate Posterior: The model and network defined so far must be trained in the direction to describe the optimal $q(z, c|\boldsymbol{x})$. The optimal variational distribution denotes the minimizer of the distance between $q(z, c|\boldsymbol{x})$ and true distribution $p(z, c|\boldsymbol{x})$, which can be represented using Kullback Leibler (KL) divergence as follows:

$$\min_{\boldsymbol{\phi}, \boldsymbol{\psi}, \{\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c, \boldsymbol{\pi}_c\}_{c=1}^{C}} \underbrace{KL\left(q_{\boldsymbol{\phi}}(z, c|\boldsymbol{x}) \middle|\middle| p_{\boldsymbol{\psi}}(z, c|\boldsymbol{x})\right)}_{\mathcal{L}_{\text{OBJ}}}. \tag{10}$$

We define the KL divergence term as $\mathcal{L}_{\text{OBJ}}$. Then the optimal variational posterior of c can be found from the objective.

Lemma 2: The optimal approximate posterior of $\boldsymbol{c}$ is given as the expectation of $p(c|z)$ over the approximate posterior of z, i.e., $q^*(\boldsymbol{c}|\boldsymbol{x}) = \mathbb{E}_{q(z|\boldsymbol{x})}[p(\boldsymbol{c}|z)]$.

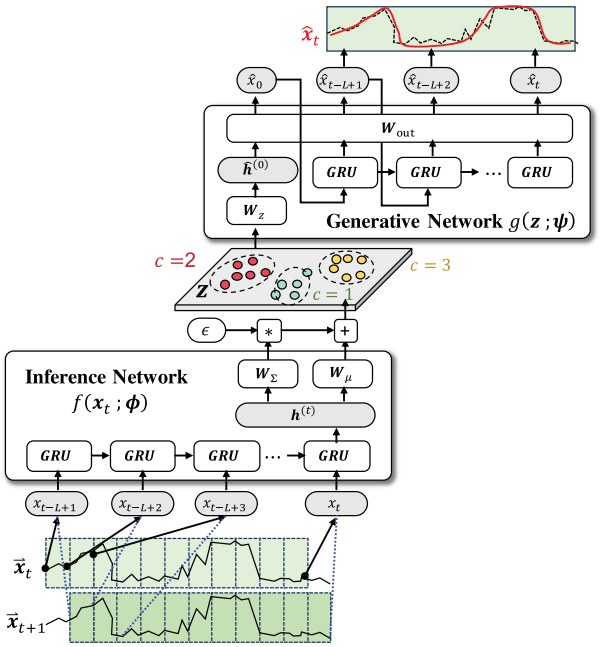


Fig. 3. Illustration of VaRDE architecture. Inference network corresponds to (8), and Generative network works as (9).

Proof of Lemma: The objective $\mathcal{L}_{\text{OBJ}}$ can be factorized as

$$\begin{aligned}\mathcal{L}_{\text{OBJ}} &= \int_z \sum_c q(z,c|\boldsymbol{x})\Big(\log\frac{q(z,c|\boldsymbol{x})}{p(z,c,\boldsymbol{x})}+\log p(\boldsymbol{x})\Big)dz\\ &= \mathbb{E}_{q_\phi(z,c|\boldsymbol{x})}\Big[\log\frac{q_\phi(z,c|\boldsymbol{x})}{p_\psi(z,c,\boldsymbol{x})}\Big]+\log p(\boldsymbol{x}).\\ \Leftrightarrow \log p(\boldsymbol{x}) &= \mathcal{L}_{\text{OBJ}}(\cdot;\boldsymbol{\phi},\boldsymbol{\psi})+\underbrace{\mathbb{E}_{q_\phi(z,c|\boldsymbol{x})}\Big[\log\frac{p_\psi(z,c,\boldsymbol{x})}{q_\phi(z,c|\boldsymbol{x})}\Big]}_{\mathcal{L}_{\text{ELBO}}}.\end{aligned}$$

Since $p(\boldsymbol{x})$ does not depend on the $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$, minimizing $\mathcal{L}_{\text{OBJ}}$ can be replaced with maximizing the latter term, usually called evidence lower bounce (ELBO); it is factorized as follows [44]:

$$\begin{aligned}\mathcal{L}_{\text{ELBO}} &= \int_z\sum_{\boldsymbol{c}} q_{\boldsymbol{\phi}}(z|\boldsymbol{x})q(\boldsymbol{c}|\boldsymbol{x})\log\Big[\frac{p_\psi(\boldsymbol{x}|z)p(z)p(\boldsymbol{c}|z)}{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})q(\boldsymbol{c}|\boldsymbol{x})}\Big]\\ &= \int_z q_\phi(z|\boldsymbol{x})\Big[\log\frac{p(\boldsymbol{x}|z)p(z)}{q_\phi(z|\boldsymbol{x})}+\sum_{\boldsymbol{c}} q(\boldsymbol{c}|\boldsymbol{x})\log\frac{p(\boldsymbol{c}|\boldsymbol{x})}{q(\boldsymbol{c}|\boldsymbol{x})}\Big]dz\\ &= \mathbb{E}_{q_\phi(z|\boldsymbol{x})}\Big[\log\frac{p_\psi(\boldsymbol{x}|z)p(z)}{q_\phi(z|\boldsymbol{x})}-KL\Big(q(\boldsymbol{c}|\boldsymbol{x})||p(\boldsymbol{c}|z)\Big)\Big].\end{aligned}$$

The optimal approximate posterior of $\boldsymbol{c}$ maximizes the $\mathcal{L}_{\text{ELBO}}$ when the latter KL divergence is zero, for the KL divergence always meets non-negative by Jensen's inequality. Therefore, $KL(q^*(\boldsymbol{c}|\boldsymbol{x})||\mathbb{E}_{q_\phi(z|\boldsymbol{x})}[p(\boldsymbol{c}|z)])=0$ holds. ∎

From Lemma 2, the optimal form of approximate posterior c can be reformulated into the computational form with the Monte-Carlo average and reparameterization trick [46]. Finally, we update Temporal Data Pool with this optimal distribution as follows.

Definition 2 (Temporal Data Pool With Variational Inference): From given multiple customers' dataset $\boldsymbol{S}\coloneqq\cup_{m\in[M]}S_m$, $\tilde{\mathcal{D}}_i$ is redefined as

$$\Big\{(\boldsymbol{x}_t,y_t)\Big|\arg\max_{c\in[C]}q^*\big(c|\boldsymbol{x}_t,\boldsymbol{\phi}^*,\boldsymbol{\mu}_c^*,\boldsymbol{\Sigma}_c^*\big)=i,(\boldsymbol{x}_t,y_t)\in\boldsymbol{S}\Big\}$$

where

$$\gamma_c^*\coloneqq q^*\big(c|\boldsymbol{x}_t,\boldsymbol{\phi},\boldsymbol{\mu}_c,\boldsymbol{\Sigma}_c\big)=\frac{1}{S}\sum_{s=1}^{S}\frac{\pi_c p\Big(\boldsymbol{z}_t^{(s)}|c\Big)}{\sum_{c=1}^{C}\pi_c p\Big(\boldsymbol{z}_t^{(s)}|c\Big)},$$

$\boldsymbol{z}_t^{(s)}=\boldsymbol{\mu}_z+\epsilon^{(s)}\odot\boldsymbol{\Sigma}_z$, $\epsilon^{(s)}\sim\mathcal{N}(0,\mathbb{I})$, $[\boldsymbol{\mu}_z,\log\boldsymbol{\Sigma}_z]=f(\boldsymbol{x}_t;\boldsymbol{\phi})$, $p(\boldsymbol{z}|c)=\mathcal{N}(\boldsymbol{z}|\mu_c,\Sigma_c)$, and S is the number of samples for the MC average.

Here, all parameters used in Definition 2 should maximize $\mathcal{L}_{\text{ELBO}}$, of which the tractable form is derived with Assumption 5 and [44, Lemma 1] as follows:

$$\begin{aligned}&\mathcal{L}_{\text{ELBO}}(\boldsymbol{x}_t)\\ &=\mathbb{E}_{q(\boldsymbol{z}_t,c|\boldsymbol{x}_t)}\big[\log p(\boldsymbol{x}_t|\boldsymbol{z}_t)\big]-KL\Big(q(\boldsymbol{z}_t,c|\boldsymbol{x}_t)\Big|\Big|p(\boldsymbol{z}_t,c)\Big)\\ &=-\alpha\frac{1}{S}\sum_{s=1}^{S}\sum_{t=1}^{L}\Big(x_t-\hat{x}_t^{(s)}\Big)^2+\sum_{c=1}^{C}\gamma_c^*\log\pi_c\\ &\quad-\frac{1}{2}\sum_{c=1}^{C}\gamma_c^*\sum_{h=1}^{H}\left[\log\Big(2\pi\Sigma_c^h\Big)+\frac{\Sigma_z^h+\big(\mu_z^h-\mu_c^h\big)^2}{\Sigma_c^h}\right]\\ &\quad+\frac{1}{2}\sum_{h=1}^{H}\Big(\log 2\pi+\log\Sigma_z^h+1\Big)-\sum_{c=1}^{C}\gamma_c^*\log\gamma_c^*,\end{aligned}\tag{11}$$

where $\hat{x}_t^{(s)}$ is a reconstructed value from $\boldsymbol{z}_t^{(s)}$ and α is a hyper parameter to give weight on reconstruction. $\mu_\cdot^h$ and $\Sigma_\cdot^h$ denote the h^{th} element of each vector, respectively. All optimal VaRDE parameters $\boldsymbol{\phi}^*$, $\boldsymbol{\psi}^*$, $\boldsymbol{\pi}_c^*$, $\boldsymbol{\Sigma}_c^*$, and $\boldsymbol{\mu}_c^*$ are obtained in a way to maximize Eq. (11). Eventually, Temporal Data Pools $\{\tilde{\mathcal{D}}_c\}_{c=1}^C$ are established, based on Definition 2.

5) Determining C of Temporal Data Pools: For the temporal data pool construction, we started from Assumption 2, where C is given as prior. It is not conceivable, yet we need to find C from $\boldsymbol{S}$.

To find the most likely parameter, we employ the Bayesian Information Criterion (BIC) [47] to measure how well C reflects the hidden embedding. To find the C minimizing the BIC, we iterate to measure the BIC for $\boldsymbol{z}$ with the given C after training VaRDE. However, finding optimal C to minimize BIC is costly. Instead, we make an early stopping when the BIC increases compared to the previous value. The rationale for early stopping is that we aim to improve computing resource efficiency by minimizing the number of models. The upper side of Fig. 4 demonstrates the actions to construct Temporal Data Pools for a given training dataset $\boldsymbol{S}$. Finally, we train each forecasting model from the given temporal data pools. For clarity, we call the step to construct optimal temporal data pools Phase 1 and the step to train forecasting models using them Phase 2.

C. Making Up for Assumption 4 With Meta-Initialization

Using the Temporal Data Pools given in Phase 1, the TDP framework trains ISTLF models corresponding to each pool distribution. One matter in the training model is unanticipated

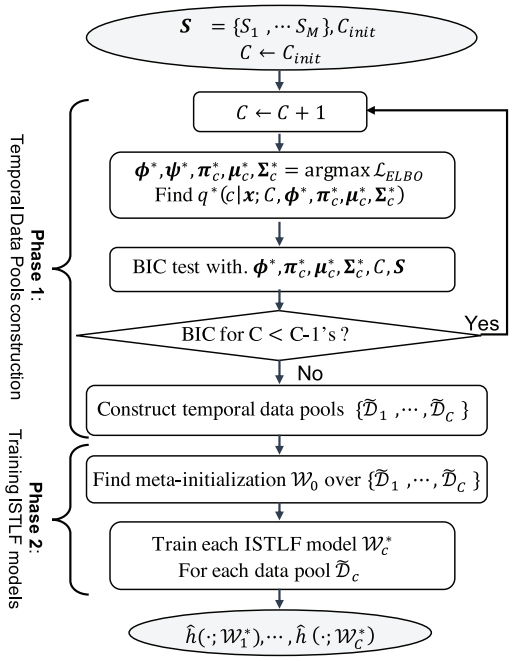


Fig. 4. Training steps of the proposed TDP framework.

data imbalance among the pools, i.e., $|\tilde{\mathcal{D}}_c| \neq |\tilde{\mathcal{D}}_k|$. Since the pools are constructed only based on the distribution inference, it may not guarantee enough data in a temporal data pool unexpectedly. This makes Assumption 4 not hold, which may increase the generalization error (5) of the TDP.

To remedy this, we additionally employ the initialization strategy of MAML [24] to TDP.[1] MAML is an optimization-based meta-learning that finds the best initial parameter across multiple tasks through gradient descent to cover data shortage situations. The effect of MAML stems from feature reuse of good initialization [48]. Thus, the meta-initialization can provide shared features over multiple tasks of the TDP, making effective adaptations with fewer data. Moreover, the meta-initial parameter serves as a prior in hierarchical empirical Bayes [49]. Recent studies also showed the MAML's effect on the generalization gap in data paucity environment with Bayesian PAC bound [50], [51], which implies this meta-initialization can complement Assumption 4.

To find the meta-initialization parameter $\mathcal{W}_0$ over C TDP tasks from Phase 1, we first split each temporal data pool into training and validation datasets, $\tilde{\mathcal{D}}_c^{tr}$ and $\tilde{\mathcal{D}}_c^{val}$. Then, the meta initializer $\mathcal{W}_0$ is found to minimize the following objective:

$$\min_{\mathcal{W}_0} \sum_{c=1}^{C} \mathcal{L}_{\tilde{\mathcal{D}}_c^{val}}\Big(\mathcal{W}_0 - \gamma \nabla_{\mathcal{W}_0} \mathcal{L}_{\tilde{\mathcal{D}}_c^{tr}}(\mathcal{W}_0)\Big), \tag{12}$$

where

$$\mathcal{L}_{\mathcal{D}_c}(\mathcal{W}) = \sum_{(\boldsymbol{x}, y) \in \mathcal{D}_c} \mathcal{L}\Big(h_c(\boldsymbol{x}, y; \mathcal{W})\Big).$$

[1]This meta-learning-based initialization will be referred to as *meta-initialization*. Unlike MAML, which uses the initial parameter in learning new unseen tasks, our TDP exploits the meta-initialization parameter for ISTLF training of temporal data pools obtained from Phase 1.

Algorithm 1: Phase 2. Training ISTLF Models With MAML

input: Temporal Data Pools $\{\tilde{\mathcal{D}}_c^{\text{TDP}}\}_{c=1}^{C}$
output: ISTLF model parameters $\{\mathcal{W}_c\}_{c=1}^{C}$
Split each data pool into $\tilde{\mathcal{D}}_c^{tr}$ and $\tilde{\mathcal{D}}_c^{val}$; Initialize $\mathcal{W}_0$
while *not converge* **do**
  for $c \in [C]$ **do**
    $\mathcal{W}_c \leftarrow \mathcal{W}_0$ $\mathcal{W}_c \leftarrow \mathcal{W}_c - \gamma \nabla_{\mathcal{W}_c} \mathcal{L}_{\tilde{\mathcal{D}}_c^{tr}}(\mathcal{W}_c)$ ▷ Inner loop
  end
  $\mathcal{W}_0 \leftarrow \mathcal{W}_0 - \beta \nabla_{\mathcal{W}_0} \sum_{c=1}^{C} \mathcal{L}_{\tilde{\mathcal{D}}_c^{val}}(\mathcal{W}_c)$ ▷ Outer loop
end
for $c \in [C]$ **do**
  $\mathcal{W}_c \leftarrow \mathcal{W}_0$ ▷ Initialize with meta-initial parameter
  while *not converge* **do**
    $\mathcal{W}_c \leftarrow \mathcal{W}_c - \gamma \nabla_{\mathcal{W}_c} \mathcal{L}_{\tilde{\mathcal{D}}_c}(\mathcal{W}_c)$ ▷ Training c model
  end
end

This optimization can be resolved through inner-loop and outer-loop optimization [24]. After getting the meta-initializer $\mathcal{W}_0$, we train each forecasting model initializing with $\mathcal{W}_0$. Algorithm 1 demonstrates this procedure of Phase 2. Fig. 4 demonstrates the overall training steps of the proposed TDP framework, including both Phase 1 and 2.

V. Experimental Evaluation & Discussion

This section explains how we evaluate our proposed work and describes its efficacy. We begin by summarizing the experiment settings, which include the dataset, model, and hyperparameters. Then we measure the efficacy of our proposal by two metrics: RMSE and MAAPE.

A. Experiment Settings

1) Dataset: We employed actual power data from 99 AMI smart meters built by Korea Electric Power Corporation for evaluation. Specifically, we exploited low-voltage meters to evaluate the TDP effect on residential ISTLF, which is trickier than system-level load forecasting due to its high uncertainties [5]. The data were collected over the course of a year, from January to December of 2016. Each smart meter measures the power consumption every 15 minutes. Hereby, the total number of samples per customer (S/C) is 35,040. We divide the dataset into two halves for evaluation: the seen and the unseen customer datasets. The 'seen' customer dataset is further split into two parts: the training and the test dataset, with a ratio of 8:2. VaRDE and ISTLF exploit the training dataset, consisting of data samples from January to the middle of October. In particular, the training dataset is randomly split into training and validation within each temporal data pool to find meta-initialization; we set the split ratio as 0.8. They are respectively used for the inner and outer loop in Algorithm 1. We only used the test dataset for the accuracy evaluation, which is never shown in both VaRDE and ISTLF models. Table I summarizes this dataset configuration.

TABLE I
SUMMARY OF DATASET USED IN THE EVALUATION

		Customers	S/C	Period (days)
Seen Customer	Training	96 (1-96)	28,032	292 (Jan. - Oct.)
	Test		7,008	73 (Oct. - Dec.)
Unseen Customer		3 (97-99)	35,040	365 (Jan. -Dec.)

2) VaRDE Network: The VaRDE consists of GRU cells, where a hidden dimension of 288, i.e., $D = 288$. Furthermore, the latent embedding size H is set to 24. Training VaRDE from scratch may result in divergence since the training is often biased to minimize the reconstruction loss while ignoring the regularization objective. To evade this phenomenon, we initialize the VaRDE with a pre-trained auto-encoder with a reconstruction loss and the same network architecture as VaDE did [44]. For training VaRDE, we implement the network with PyTorch and the Adam optimizer with a learning rate of 1e-5.

3) Forecasting Network: We exploit existing network architecture to demonstrate that the proposed TDP framework is a model-agnostic for the ISTLF, providing a fair comparison with other methods. We adopt the LSTM-based forecasting network proposed by Kong et al. [5] as the ISTLF model trained in Phase 2. The network consists of two LSTM layers, whose hidden size H is 24. The input vector contains a sequential vector of AMI power data, a one-hot vector representing the day information, and the other one-hot vector for the time index. The lookback size L of the input vector is set to 96, i.e., one day; look forward step T is set to 1, the same as Kong's model. For phase 2, we construct and train the ISTLF models with TensorFlow and Keras with Adam optimizer and a learning rate of 0.001.

In particular, we apply the proposed TDP framework for both point and Bayesian forecasting. These extensive test cases highlight our novelty, wherein the TDP framework outperforms both cases due to its distribution inference-based pooling. For point forecasting, we used a similar evaluation method as Yang et al. except for not using holiday mark as an input; for Bayesian forecasting, we adopted MC dropout as most of the pooling-based stochastic ISTLF have done [18], [33]. We tested the average forecasting value when the prior probability was set to 0.3 to compare forecasting accuracy for the Bayesian case. The number of MC-average samples S was set to 20.

4) Competing Methods: For evaluation, we implemented six competing methods and two proposed methods. We constructed all frameworks with the identical ISTLF network architecture as described earlier. For two types of evaluation, point and Bayesian forecasting, two forecasting models are trained respectively: one is a basic ISTLF network, and the other is an ISTLF network with dropout layers. The following are descriptions of all frameworks that have been trained for this evaluation:

- *Global (One-to-all) [16]:* It trains a multi-task learning model with all customer data.
- *Local (One-to-One) [5]:* It builds a forecasting network for each customer.
- *Local with Meta-Init (One-to-One)[15]:* It trains the independent network with MAML. Since [15] applied for a few-shot learning case only, we applied MAML the same as ours for a fair comparison.
- *CDP-Random [17]:* It generates pools at random and trains the models for each pool, where a model takes the user identifier as an input.
- *CDP-AHC [7], [18]:* The ensemble method clusters the average load profiles of each customer using Agglomerative hierarchical clustering (AHC), and the CDP-based forecasting models are constructed based on the result.
- *CDP-kMeans [19]:* The CDP-based forecasting, where pools are constructed using k-means clustering.
- *(TDP with Random Init) TDP framework with random initialization:* Our proposed TDP framework without the meta-initialization in Phase 2.
- *(TDP with Meta-Init):* Our proposed TDP framework with the meta-initialization (Algorithm 1) in Phase 2.

B. Overall Evaluation

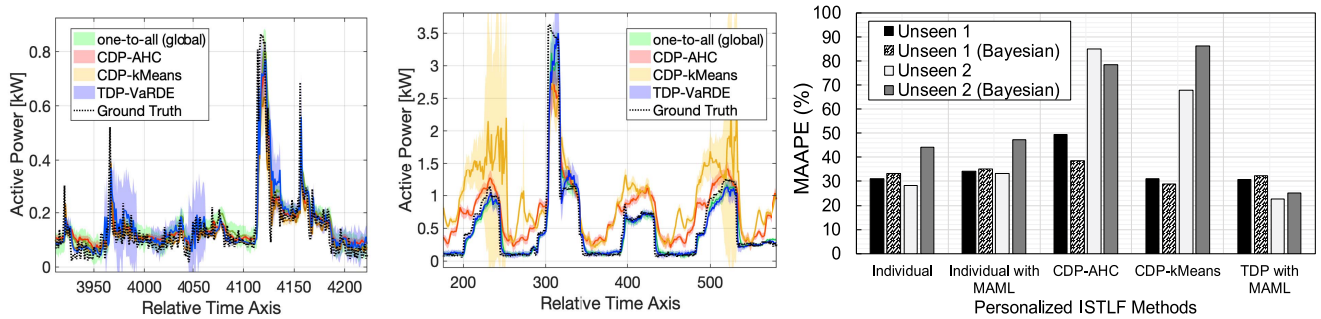
1) Description of Evaluation Results: For the experimental analysis, we used two metrics to measure the accuracy of the competing methods and our proposal: RMSE and MAAPE [52]. The MAAPE is a metric devised for AMI data with many near-zero values. Table II shows the evaluation results, which include the average results for multiple customers — 96 for *seen customers* and three *unseen customers* at training time. Since each customer has different scales, we averaged the RMSE values after normalizing over competing methods of a customer for a fair comparison. Simple average RMSE and MAAPE are insufficient to evaluate its effect, so we count the number of customers for whom the approach has the top one accuracy. The number in parentheses denotes the counts.

The table contains forecasting results from two methods: point forecasting (a $\sim$ h) and Bayesian forecasting ($i \sim p$). The only difference between point forecasting and Bayesian forecasting is whether MC-dropout is used [18], [35]. Among the methodologies, *b)* $\sim$ *h)* offers customized ISTLF. On the other hand, the one-to-all approach trains one model for all customers, resulting in the lowest accuracy due to the distribution discrepancy among customers. *b)* and *c)* train a forecasting network for each customer, with each network tailored to a target customer. Thus, they show higher accuracy, but the number of forecasting models is the same as the number of customers, 96. *d)* to *h)* are pooling-based ISTLF that moderate between one-to-all and one-to-one, i.e., one-to-multiple. *d)* to *f)* are customer data pooling (CDP) methods that train fewer models than the number of customers. Their accuracy depends on how the pools are constructed. *g)* to *h)* are our TDP methods in which only the initialization methods differ.

2) Forecasting Accuracy for Seen Customers: In terms of mean RMSE and MAAPE, our proposal (*g, h, o*, and *p*) outperforms all competing methods for both point and stochastic forecasting. It improves accuracy from 9.79% up to 79.13% in RMSE. In addition, the proposed TDP frameworks have the

TABLE II
SUMMARY OF THE EVALUATION RESULT. FORECASTING ACCURACIES WERE MEASURED WITH TWO METRICS: RMSE AND MAAPE. EACH ITEM DENOTES AVERAGE VALUE OVER THE CUSTOMERS — 96 AND 3 FOR VALIDATION AND TEST, RESPECTIVELY. ABOUT THE RMSE, THE AVERAGE VALUE OF NORMALIZED OVER CLIENTS WAS COMPUTED FOR EVEN COMPARISON. THE VALUE IN THE PARENTHESIS DENOTES THE NUMBER OF CASES WHERE THE METHOD WINS THE TOP-ONE ACCURACY COMPARED TO OTHERS

Method			Models	Seen Customer / Test Data		Unsee Customer / Test Data	
				RMSE (/96)	MAAPE [%] (/96)	RMSE (/3)	MAAPE [%] (/3)
Point Forecasting	a)	Global (One-to-All) [15]	1	0.618 (0)	52.37 (0)	0.285 (0)	58.4 (0)
	b)	Local (One-to-One) [5]	96	0.143 (21)	38.63 (22)	N/A	N/A
	c)	Local with Meta-Init. [15]	96	0.169 (1)	42.62 (3)	N/A	N/A
	d)	CDP-Random [17]	9	0.283 (14)	41.33 (10)	N/A	N/A
	e)	CDP-AHC [18]	32	0.160 (13)	38.76 (9)	0.483	70.1
	f)	CDP-kMeans [19]	24	0.168 (6)	39.41 (10)	0.194	46.7
	g)	**TDP with Random Init (proposed)**	24	0.131 (18)	38.86 **(28)**	0.154	40.19
	h)	**TDP with Meta-Init. (proposed)**	24	**0.129 (23)**	**38.41** (14)	**0.136 (3)**	**37.02 (3)**
[35] Bayesian Forecasting (p=0.3)	i)	Global (One-to-All)	1	0.373 (0)	45.77 (1)	0.199	49.42
	j)	Local (One-to-One)	96	0.165 (5)	42.48 (6)	N/A	N/A
	k)	Local with Meta-Init.	96	0.165 (2)	42.96 (5)	N/A	N/A
	l)	CDP-Random	9	0.329 (10)	48.69 (7)	N/A	N/A
	m)	CDP-AHC	32	0.190 (7)	43.03 (8)	0.234	49.42
	n)	CDP-kMeans	24	0.179 (24)	42.76 (12)	0.278	55.23
	o)	**TDP with Random Init. (proposed)**	24	0.139 (7)	41.35 (14)	0.170	43.339
	p)	**TDP with Meta-Init. (proposed)**	24	**0.131 (41)**	**39.74 (43)**	**0.154 (3)**	**40.50 (3)**



(a) Unseen 1 (b) Unseen 2 (c) Unseen Comparison

Fig. 5. Forecasting result for two *Unseen* customers.

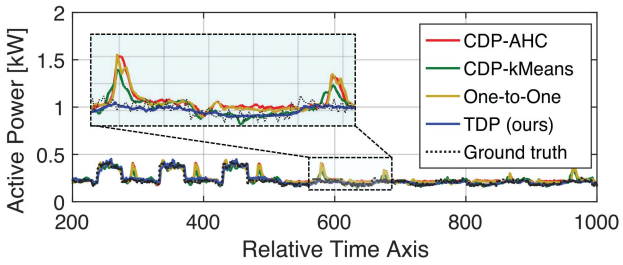
best accuracy counts written in parenthesis; it has about half of the best forecasting accuracy, including two initializations.

Interestingly, Bayesian forecasting's accuracy is lower than point forecasting because the uncertainty probability is given as 0.3. Nevertheless, our proposal shows a low error rate in the prediction mean value because the TDP framework aligns the distribution of each task based on distribution inference. The significance of distribution matching can be demonstrated by comparing two MAML initialization cases: one-to-one (*b* and *k*) and the TDP (*h* and *p*). When comparing *b* and *c*, the one-to-one with meta-initialization gets worse in forecasting accuracy; however, our TDP benefits from MAML compared to *g* and *h*. We conjecture that this is because the TDP makes training data for each task to be i.i.d. samples, meeting the MAML assumption. To further evaluate the resource efficiency and scalability of TDP, we also employed the number of models used in ISTLF for more than 96 customers as a metric. As a result, the TDP shows the lowest error with 24 forecasting models, much less than the *one-to-one* methods. Furthermore, overheads caused by dynamic model assignment can be canceled by using the StreamDL serving platform [53], providing pipelining between data pool inference and forecasting.

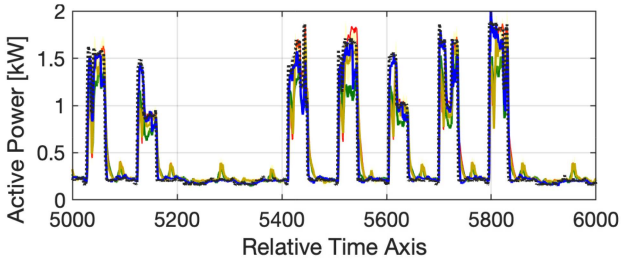
3) Forecasting Accuracy for Unseen Customer: The unseen customer case clearly demonstrates the advantages of the TDP framework. Our proposal provides robust forecasting to new customers without the need for additional training. The lowest average MAAPEs for point and Bayesian forecastings are 37.02% and 40.50%, respectively. Furthermore, it leads to an error gap of up to 33% compared to the worst case. Fig. 5 depicts the forecasting results for two Unseen customers. The CDP k-Means performs well in Unseen customer 1, but its predictive performance is comparatively lower for Unseen customer 2, the same as the CDP-AHC. Note that the one-to-one methods in Fig. 5(c) have undergone additional training. The one-to-one methods require time to collect enough training data for each customer and computational resources for additional training. The figure shows that our proposal outperforms one-to-one forecasting.

C. Effect of the TDP Framework on Concept Drift

The TDP framework was designed to handle concept drift with a pooling strategy. In contrast to the existing pooling strategy, which statically assigns a forecasting network to a



(a) From 200 to 1000



(b) From 5000 to 6000

Fig. 6. Forecasting comparison plot for AMI 66's test data.

TABLE III
MAAPE COMPARISON AMONG FOUR POOLING METHODS FOR THE DISTRIBUTION-CHANGE CASE

	CDP-Random	CDP-AHC	CDP-kMeans	TDP
Point	13.22	12.07	15.25	**11.56**
Stochastic	61.56	17.38	16.02	**10.23**

customer, our proposed TDP dynamically selects the most probable forecasting model. To evaluate the efficacy of the TDP when concept drift occurs, we present a case in which the data distribution is changed in test time.

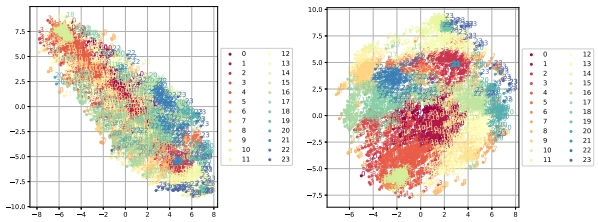
As Fig. 6 demonstrates the distribution change case, Fig. 6a and 6b show different times for the same smart meter. At first, its power load was less than 0.5, and most power values were nearly zero. As time went by, the active power rose by almost 2 kW. We can notice that the distribution of the customer has shifted at the test time. Because of this change in distribution, competing methods did not respond to nearly zero values. Fig. 6a illustrates how the methods predicted in the near-zero interval. On the other hand, our proposed method exhibits accurate prediction in both the activated sections (from 5400 to 5500) and the section close to zero. Table III summarizes the MAAPE of the point and stochastic forecasting of three CDP methods and our proposal. One interesting fact is that the existing approaches increase the error in Bayesian forecasting, whereas our method improves the accuracy. The distribution matching technique of the TDP increases the stochastic forecasting accuracy by providing an additional prior of the current input in such highly stochastic situations.

D. Ablation Study

We conducted an ablation study to emphasize the effect of distribution-aware factors of the proposal, as shown in Table IV. Here, the TDP denotes the way to construct pools over sequence chunks as described in Definition 1, except

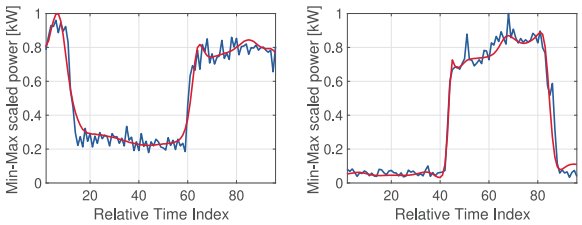
TABLE IV
ABLATION STUDY FOR EFFECT ON BAYESIAN FORECASTING

Method			MAAPE (%)	
Pooling Method		Initialization	Seen	Unseen
CDP	**k-Means**	**Random**	42.76	55.23
TDP	**k-Means**	**Random**	41.03	44.76
	VaRDE	**Random**	40.93	42.36
		MAML	**39.74**	**40.50**



(a) tSNE (b) VaRDE + tSNE

Fig. 7. Plot using tSNE to check clustering effect of VaRDE.



(a) Seen customer (b) Unseen customer

Fig. 8. Reconstruction result of VaRDE.

for how to infer the distribution, i.e., VaRDE. It aligns data pools, so the TDP shows better accuracy than that of the customer data pooling under the same clustering and initialization method, i.e., 42.76% vs. 41.03%. Its effect is more remarkable for the Unseen customer case with a more considerable margin, where the forecasting error is reduced by 10.47%.

Besides that, we also compare the TDP with K-Means and the TDP with VaRDE (Definition 2) to validate the influence of VaRDE on distribution awareness. The TDP with VaRDE improves accuracy for both seen and unseen customers, with MAAPEs of 41.03% vs. 40.93% and 44.76% vs. 42.36%, respectively. Because VaRDE clusters by distribution inference, it is better suited for matching individual loads with the high stochastic distribution. In addition, Fig. 7 and 8 show how well the proposed VaRDE performs to cluster individual load profiles. Fig. 7 demonstrates that VaRDE extracts good embedding to be clustered by dimension reduction compared to the raw load profiles. Moreover, it not only extracts the features to be clustered but also represents the distribution of profiles shown through the reconstruction plot in Fig. 8.

Finally, the meta-initialization highly improves the accuracy in the stochastic ISTLF by reducing MAAPE from 40.93% to 39.74% and from 42.36% to 40.50% in Seen and Unseen cases, respectively. The feature reuse through initialization obtained by MAML compensates for the pool deviation and improves accuracy. One interesting point is that the TDP

TABLE V
EFFECT OF META-INITIALIZATION: MAAPE(%) COMPARISONS. WE COMPARED EFFICACY OF META-INITIALIZATION ON ONE-TO-ONE FORECASTING AND THE PROPOSED TDP FRAMEWORK

Initialization	Seen Customer		Unseen Customer	
	one-to-one	TDP	one-to-one	TDP
Random	42.48	41.35	44	44.36
Meta-Initialization	42.96	**39.74**	48.19	**40.5**

increases the gain from MAML compared to the one-to-one ISTLF. The proposed framework redefines tasks in order to match the distribution of each task. Table V shows that TDP improves the efficacy of MAML, whereas the on-to-one is not improved by MAML because each customer's data contains data samples from different distributions due to its non-stationary property.

VI. CONCLUSION

Due to its nature and customer-specific data distribution, individual load forecasting is highly stochastic and heterogeneous. As a result, the forecasting model must be trained in a distribution-aware manner. Existing pooling-based individual load forecasting approaches could overcome overfitting issues; however, they only consider distribution discrepancies between inter-customers, not intra-customers. Therefore, in this study, we proposed a new pooling method called the TDP, which constructs a data pool over both customer and temporal axes based on the distribution inference of each sample. We especially presented a theoretical analysis of the effect of the TDP based on statistical learning theory; to put this theoretical paradigm into practice, we proposed the probabilistic deep embedding method with recurrent networks called VaRDE. In addition, we raised the possible issue of data imbalance among the pools when naively using the TDP; we proposed to adopt meta-initialization to the TDP.

The TDP, which performs pooling solely based on distribution inference for all input data, shows robust forecasting results even under distribution changes, as expected. Moreover, this TDP framework demonstrates highly accurate forecasting for a new customer without additional training. The TDP achieves such excellent forecasting performance because it infers the distribution at each point and determines the forecasting model based on distribution inference. This approximates the hierarchical probabilistic model that determines the model's condition. By doing so, our Bayesian forecasting studies also confirmed that it has higher accuracy by providing different priors according to distribution for each model. In particular, the TDP allows resource-efficient inference forecasting in a multi-client cloud environment via model sharing. Therefore, it is anticipated that this technique can be used for intelligent power management of customers in such a Cloud environment.

REFERENCES

[1] M. Imani and H. Ghassemian, "Residential load forecasting using wavelet and collaborative representation transforms," *Appl. Energy*, vol. 253, Nov. 2019, Art. no. 113505.

[2] S. N. Fallah, M. Ganjkhani, S. Shamshirband, and K.-W. Chau, "Computational intelligence on short-term load forecasting: A methodological overview," *Energies*, vol. 12, no. 3, p. 393, 2019.

[3] N. Ding, C. Benoit, G. Foggia, Y. Bésanger, and F. Wurtz, "Neural network-based model design for short-term load forecast in distribution systems," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 72–81, Jan. 2016.

[4] Y. Zhang, T. Huang, and E. F. Bompard, "Big data analytics in smart grids: A review," *Energy Inform.*, vol. 1, no. 1, pp. 1–24, 2018.

[5] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on LSTM recurrent neural network," *IEEE Trans. Smart Grid*, vol. 10, no. 1, pp. 841–851, Jan. 2019.

[6] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo, and Y. Xu, "Short-term residential load forecasting based on resident behaviour learning," *IEEE Trans. Power Syst.*, vol. 33, no. 1, pp. 1087–1088, Jan. 2018.

[7] Y. Wang, Q. Chen, M. Sun, C. Kang, and Q. Xia, "An ensemble forecasting method for the aggregated load with subprofiles," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3906–3908, Jul. 2018.

[8] B. Stephen, X. Tang, P. R. Harvey, S. Galloway, and K. I. Jennett, "Incorporating practice theory in sub-profile models for short term aggregated residential load forecasting," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1591–1598, Jul. 2017.

[9] X. Qiu, P. N. Suganthan, and G. A. J. Amaratunga, "Ensemble incremental learning random vector functional link network for short-term electric load forecasting," *Knowl.-Based Syst.*, vol. 145, pp. 182–196, Apr. 2018.

[10] M. Chaouch, "Clustering-based improvement of nonparametric functional time series forecasting: Application to intra-day household-level load curves," *IEEE Trans. Smart Grid*, vol. 5, no. 1, pp. 411–419, Jan. 2014.

[11] K. Yan, W. Li, Z. Ji, M. Qi, and Y. Du, "A hybrid LSTM neural network for energy consumption forecasting of individual households," *IEEE Access*, vol. 7, pp. 157633–157642, 2019.

[12] L. Jiang, X. Wang, W. Li, L. Wang, X. Yin, and L. Jia, "Hybrid multitask multi-information fusion deep learning for household short-term load forecasting," *IEEE Trans. Smart Grid*, vol. 12, no. 6, pp. 5362–5372, Nov. 2021.

[13] E. Yang and C.-H. Youn, "Individual load forecasting for multi-customers with distribution-aware temporal pooling," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2021, pp. 1–10.

[14] Y.-H. Hsiao, "Household electricity demand forecast based on context information and user daily schedule analysis from meter data," *IEEE Trans. Ind. Informat.*, vol. 11, no. 1, pp. 33–43, Feb. 2015.

[15] E. Lee and W. Rhee, "Individualized short-term electric load forecasting with deep neural network based transfer learning and meta learning," *IEEE Access*, vol. 9, pp. 15413–15425, 2021.

[16] P. Montero-Manso and R. J. Hyndman, "Principles and algorithms for forecasting groups of time series: Locality and globality," *Int. J. Forecast.*, vol. 37, no. 4, pp. 1632–1653, 2021.

[17] H. Shi, M. Xu, and R. Li, "Deep learning for household load forecasting—A novel pooling deep RNN," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 5271–5280, Sep. 2018.

[18] Y. Yang, W. Li, T. A. Gulliver, and S. Li, "Bayesian deep learning-based probabilistic load forecasting in smart grids," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4703–4713, Jul. 2020.

[19] F. Han, T. Pu, M. Li, and G. Taylor, "Short-term forecasting of individual residential load based on deep learning and K-means clustering," *CSEE J. Power Energy Syst.*, vol. 7, no. 2, pp. 261–269, Mar. 2021.

[20] H. Zang et al., "Residential load forecasting based on LSTM fusing self-attention mechanism with pooling," *Energy*, vol. 229, Aug. 2021, Art. no. 120682.

[21] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," 2020, *arXiv:2002.10619*.

[22] R. K. Jagait, M. N. Fekri, K. Grolinger, and S. Mir, "Load forecasting under concept drift: Online ensemble learning with recurrent neural network and ARIMA," *IEEE Access*, vol. 9, pp. 98992–99008, 2021.

[23] M. N. Fekri, H. Patel, K. Grolinger, and V. Sharma, "Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network," *Appl. Energy*, vol. 282, Jan. 2021, Art. no. 116177.

[24] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[25] C. Keerthisinghe, G. Verbič, and A. C. Chapman, "A fast technique for smart home management: ADP with temporal difference learning," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 3291–3303, Jul. 2018.

[26] A. Pratt, D. Krishnamurthy, M. Ruth, H. Wu, M. Lunacek, and P. Vaynshenk, "Transactive home energy management systems: The impact of their proliferation on the electric grid," *IEEE Electrific. Mag.*, vol. 4, no. 4, pp. 8–14, Dec. 2016.
[27] T. Morstyn, N. Farrell, S. J. Darby, and M. D. McCulloch, "Using peer-to-peer energy-trading platforms to incentivize prosumers to form federated power plants," *Nat. Energy*, vol. 3, no. 2, pp. 94–101, 2018.
[28] X. Sun et al., "An efficient approach to short-term load forecasting at the distribution level," *IEEE Trans. Power Syst.*, vol. 31, no. 4, pp. 2526–2537, Jul. 2016.
[29] R. Li, C. Gu, F. Li, G. Shaddick, and M. Dale, "Development of low voltage network templates—Part II: Peak load estimation by clusterwise regression," *IEEE Trans. Power Syst.*, vol. 30, no. 6, pp. 3045–3052, Nov. 2015.
[30] P. Zhang, X. Wu, X. Wang, and S. Bi, "Short-term load forecasting based on big data technologies," *CSEE J. Power Energy Syst.*, vol. 1, no. 3, pp. 59–67, Sep. 2015.
[31] F. L. Quilumba, W.-J. Lee, H. Huang, D. Y. Wang, and R. L. Szabados, "Using smart meter data to improve the accuracy of intraday load forecasting considering customer behavior similarities," *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 911–918, Mar. 2015.
[32] T.-Y. Kim and S.-B. Cho, "Predicting residential energy consumption using CNN-LSTM neural networks," *Energy*, vol. 182, pp. 72–81, Sep. 2019.
[33] K. Chen, K. Chen, Q. Wang, Z. He, J. Hu, and J. He, "Short-term load forecasting with deep residual networks," *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 3943–3952, Jul. 2019.
[34] Y. Wang, D. Gan, M. Sun, N. Zhang, Z. Lu, and C. Kang, "Probabilistic individual load forecasting using pinball loss guided LSTM," *Appl. Energy*, vol. 235, no. October 2018, pp. 10–20, Feb. 2019.
[35] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.
[36] E. Eskandarnia, H. M. Al-Ammal, and R. Ksantini, "An embedded deep-clustering-based load profiling framework," *Sustain. Cities Soc.*, vol. 78, Mar. 2022, Art. no. 103618.
[37] Y. Wang, Q. Chen, C. Kang, M. Zhang, K. Wang, and Y. Zhao, "Load profiling and its application to demand response: A review," *Tsinghua Sci. Technol.*, vol. 20, no. 2, pp. 117–129, 2015.
[38] C. Si, S. Xu, C. Wan, D. Chen, W. Cui, and J. Zhao, "Electric load clustering in smart grid: Methodologies, applications, and future trends," *J. Modern Power Syst. Clean Energy*, vol. 9, no. 2, pp. 237–252, 2021.
[39] J. Bedi and D. Toshniwal, "Empirical mode decomposition based deep learning for electricity demand forecasting," *IEEE Access*, vol. 6, pp. 49144–49156, 2018.
[40] T. Teeraratkul, D. O'Neill, and S. Lall, "Shape-based approach to household electric load curve clustering and prediction," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 5196–5206, Sep. 2018.
[41] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2018.
[42] C. Lee, S.-H. Kim, and C.-H. Youn, "Cooperating edge cloud-based hybrid online learning for accelerated energy data stream processing in load forecasting," *IEEE Access*, vol. 8, pp. 199120–199132, 2020.
[43] Y. Wang, Q. Chen, T. Hong, and C. Kang, "Review of smart meter data analytics: Applications, methodologies, and challenges," *IEEE Trans. Smart Grid*, vol. 10, no. 3, pp. 3125–3148, May 2019.
[44] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 1965–1972.
[45] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," 2014, *arXiv:1409.1259*.
[46] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.
[47] B. Stephen, A. J. Mutanen, S. Galloway, G. Burt, and P. Järventausta, "Enhanced load profiling for residential network customers," *IEEE Trans. Power Del.*, vol. 29, no. 1, pp. 88–96, Feb. 2014.
[48] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? towards understanding the effectiveness of MAML," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–21.
[49] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, "Recasting gradient-based Meta-learning as hierarchical Bayes," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–13.
[50] N. Ding, X. Chen, T. Levinboim, S. Goodman, and R. Soricut, "Bridging the gap between practice and pac-bayes theory in few-shot meta-learning," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 29506–29516.
[51] Q. Chen, C. Shui, and M. Marchand, "Generalization bounds for Meta-learning: An information-theoretic analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 25878–25890.
[52] S. Kim and H. Kim, "A new metric of absolute percentage error for intermittent demand forecasts," *Int. J. Forecast.*, vol. 32, no. 3, pp. 669–679, 2016.
[53] E. Yang, C. Lee, J.-H. Kim, T. M. Tao, and C.-H. Youn, "StreamDL: Deep learning serving platform for AMI stream forecasting," in *Proc. Int. Conf. Data Min. Workshops (ICDMW)*, 2020, pp. 723–728.

Eunju Yang (Student Member, IEEE) received the B.S. degree from the School of Computer Science and Electronic Engineering, Handong Global University, Pohang, South Korea, in 2016, and the M.S. degree from the School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2018, where she is currently pursuing the Ph.D. degree. Her research interests include distributed machine learning, deep learning training platform, deep learning serving platform, personalization, domain adaptation, continual learning, and deep-learning-based smart grid applications.

Chan-Hyun Youn (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronics engineering from Kyungpook National University, Daegu, South Korea, in 1981 and 1985, respectively, and the Ph.D. degree in electrical and communications engineering from Tohoku University, Japan, in 1994. From 1986 to 1997, he was the Head of High-Speed Networking Team, KT Telecommunications Network Research Laboratories. Since 1997, he has been a Professor with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. He was an Associate Vice-President of the Office of Planning and Budgets in KAIST from 2013 to 2017. He is also the Director of Grid Middleware Research Center and XAI Acceleration Technology Research Center, KAIST, where he is developing core technologies that are in the areas of high performance computing, explainable AI system, satellite imagery analysis, and satellite onboard computing with deep learning acceleration system. He was selected to the inaugural class of IEEE Computer Society Distinguished Contributor in 2021. He was the General Chair for the 6th EAI International Conference on Cloud Computing (Cloud Comp 2015), KAIST, in 2015. He wrote a book titled *Cloud Broker and Cloudlet for Workflow Scheduling* (Springer, 2017). He was also a Guest Editor of IEEE WIRELESS COMMUNICATIONS in 2016, and served many international conferences as a TPC member.