IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 16, 2020, accepted January 18, 2021, date of publication January 21, 2021, date of current version January 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3053317

Individualized Short-Term Electric Load Forecasting With Deep Neural Network Based Transfer Learning and Meta Learning

EUNJUNG LEE AND WONJONG RHEE, (Fellow, IEEE)
Department of Transdisciplinary Studies, Seoul National University, Seoul 08826, South Korea

Corresponding author: Wonjong Rhee (wrhee@snu.ac.kr)

This work was supported in part by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean government [20ZR1100, Core Technologies of Distributed Intelligence Things for Solving Industry and Society Problems] and in part by the Korea Medical Device Development Fund' grant funded by the Korea government (the Ministry of Science and ICT, the Ministry of Trade, Industry and Energy, the Ministry of Health Welfare, Republic of Korea, the Ministry of Food and Drug Safety) (Project Number: 202013B14).

ABSTRACT While the general belief is that the best way to predict electric load is through individualized models, the existing studies have focused on one-for-all models because the individual models are difficult to train and require a significantly larger data accumulation time per individual. In recent years, applying deep learning for forecasting electric load has become an important research topic but still one-for-all has been the main approach. In this work, we adopt transfer learning and meta learning that can be smoothly integrated into deep neural networks, and show how a high-performance individualized model can be formed using the individual's data collected over just several days. This is made possible by extracting the common patterns of many individuals using a sufficiently large dataset, and then customizing each individual model using the specific individual's small dataset. The proposed methods are evaluated over residential and non-residential datasets. When compared to the conventional methods, the meta learning model shows 7.84% and 15.07% RMSE improvements over the residential and non-residential datasets, respectively. Our results suggest that the individualized models can be used as effective tools for many short-term load forecasting tasks.

INDEX TERMS Deep learning, individualized models, meta learning, transfer learning, short-term load forecasting.

I. INTRODUCTION

Reliable energy load forecasting is an important aspect of effective energy management. Based on the forecast period, energy load forecasting can be classified into long-term, medium-term, and short-term forecasting [1]. Long-term forecasting helps in decision making for financial and operational planning. In contrast, medium- to short-term forecasting can be useful for improving power system operations such as short-term capacity scheduling or real-time controlling. Short-Term Load Forecasting (STLF) is especially useful for reliable and efficient operation of electric grids [2], and studies on STLF have typically focused on one-for-all models. One-for-all models learn from the data of all customers and infer loads for target dates using a common model for all. In contrast, individualized models are trained on the data specific to each customer or customer group and provide direct forecasting corresponding to individual customers respectively. Although individualized forecasting has the potential to fit each customer's specific load patterns, traditionally it has not been prioritized due to the lack of data and the chance of over-fitting. Lack of data has been known as one of the major failure reasons in load forecasting [3], [4]. However, we still need individualized load forecasting because residential customers have dynamic and volatile loads [5] and non-residential customers have widely different load characteristics according to their facilities and type of businesses [4]. Furthermore, smart grid infrastructures are increasingly complicating the individual load characteristics [6].

Traditional machine learning models have achieved great success in the area of STLF. Auto-Regressive Integrated Moving Average (ARIMA) has been an essential technique

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Sharif.

for decades, where the signals captured from the past values are extrapolated into the future values. More sophisticated machine learning models have been popular as well. In a case study involving office buildings [7], a Support Vector Machine (SVM) model performed better than a conventional neural network model with three layers. Fan and Chen [8] suggested a hybrid network with self-organized map (SOM) and SVM. Hong *et al.* [9] proposed a Support Vector Regression (SVR) model that predicts monthly loads using a seasonal adjustment mechanism and a chaotic immune algorithm. Clustering based models have been successfully applied, too. Al-Qahtani and Crone [10] suggested a multivariate *k*-Nearest Neighbor (KNN) regression model. Zhang *et al.* [11] used decision tree based clustering where the load of customers in each cluster is predicted by a piece-wise linear regression model. Following the success of traditional machine learning, deep learning models have become popular because they allow modeling through more complex functions and provide a potential to improve the performance. Ding *et al.* [12] suggested Multi-Layer Perceptron (MLP) model for STLF of medium-voltage and low-voltage substations. Similarly, Ribeiro *et al.* [4] also used an MLP model, and they proposed transfer learning with temporal adjustments. Kong *et al.* [5] developed two Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) cells to predict short term load. A Convolutional Neural Network (CNN) based model was suggested in [13], and it exceeded the performance of LSTM models and ARIMA with exogenous inputs. Choi *et al.* [14] combined a Residual Network (ResNet), which is an advanced type of CNN, with LSTM units. The combined model showed the best performance among MLP, LSTM, and ResNet models in their experiment. A modified ResNet without LSTM outperformed wavelet neural network models in [15].

Even though many machine learning and deep learning models have been studied, a clear limitation has been that almost all have considered one-for-all models instead of individualized models. A customer's electric load pattern, however, might have unique and multi-faceted characteristics where many different features contribute toward the final pattern. If we consider all possible combinations of the features, the number of possible electric load patterns can be extremely large. The number of unique features, however, might not be very large and a person skilled in the art might be able to tell much about a customer's electric load pattern simply by looking at the customer's recent electric load data. For instance, residential customers might be well characterized by looking at the last several days of electric load patterns. A customer might always have very low usage in the weekday afternoon while another customer might have varying usage in the weekday afternoon, depending on how many family members are present at home each day. In such a residential scenario, it can be crucial to identify the common and important features by investigating the data of many residential customers, to recognize which features are present and important for the customer whose electric load needs to be predicted, and to use the information in an individualized way such that the future load prediction can be improved. Obviously, one-for-all models are limited because the algorithms are not specialized for individualization. Clustering models are also limited in that they do not focus on learning the features. Instead, they simply try to group the individual patterns that are formed as the combination of the underlying features.

There have been a few approaches for making individualized models work. Tascikaraoglu and Sanandaji [16] utilized external information. They employed a spatio-temporal algorithm using a multivariate autoregressive model with the data from a variety of electrical appliances in the target house and surrounding houses. The spatial information from the surrounding houses significantly improved the prediction accuracy at the individual level. The performance, however, was verified only for the case of two residential customers. Because gathering sufficiently useful external data can be practically impossible for large scale deployments, algorithmic approaches are strongly preferred. Mocanu *et al.* [17] proposed reinforcement transfer learning with deep belief networks, where the model was trained with six years of data and evaluated over five buildings. Ribeiro *et al.* [4] developed a transfer learning model that has two adaptation phases for transferring temporal knowledge and non-temporal features. The developed model with MLP performed slightly better than the SVM model when evaluated over four buildings. Both transfer learning studies were for building load forecasting, and the small number of buildings might have been a critical limitation. For transfer learning to work well, it is crucial to construct a base learner that can serve as a good initialization point of the following individualization. But when only a few buildings are used for training the base learner, there is a high chance that the characteristics of the target building are quite different from the characteristics of the buildings used for constructing base learner.

In this work, we propose to use transfer learning [18] and meta-learning [19] for individualized STLF, especially for the tasks with a large number of individuals as in residential STLF problems. Whereas the traditional models train an individualized model using the data of the target individual only, the two methods aim to use the knowledge from many other individuals as well. Despite the existing works on applying transfer learning in energy areas, to the best of our knowledge, we are the first to apply transfer learning with an explicit focus on the STLF with a large number of individuals. In our experiments, the number of customers for training the base learner is 2,633 for the residential dataset and 265 for the non-residential dataset. For meta learning, we propose a meta learning framework with a few-shot mechanism, named as *few-shot short-term load forecasting*. In the deep learning algorithm society, meta learning has received great attention in the last few years [19]–[25]. With our best knowledge, this is the first work to adopt the latest meta learning techniques to STLF tasks. In the energy areas, the suggested framework can be applied to a variety of tasks that are based on time-series

datasets. The details of transfer learning and meta learning are provided in Section II.

The rest of the paper is organized as follows. Section II provides the background of transfer learning and meta learning. Also, a brief introduction is provided on the deep learning models that are used as the base models for integrating transfer learning and meta learning. Section III presents traditional, transfer learning, and meta learning frameworks for individualized STLF. Section IV describes the experimental setting for the frameworks traditional learning, transfer learning, and meta learning approaches. The performance of those three approaches, together with statistical, ARIMA, and one-for-all learning approaches, is compared and discussed in Section V. In addition, we analyze algorithm performance over seven months to investigate the seasonal effects on the algorithm accuracy and provide experiment results for cold-start where only a limited amount of past data is available. Section VI summarizes the key results and concludes the paper.

II. BACKGROUND

We start by describing transfer learning and meta learning. We also present deep learning models that are used as the base for implementing transfer learning and meta learning.

A. TRANSFER LEARNING

In machine learning, a task can be loosely defined as a dataset and loss function $\mathcal{T} = \{\mathcal{D}, \mathcal{L}\}$. Transfer learning is intended to improve learning of a specific task (target task) using the knowledge from different but related tasks (source tasks) [18]. With transfer learning, we can extract the knowledge from the source tasks and use it for an improvement in the target task.

Consider a situation where we have a source task $\mathcal{T}_s$ and a target task $\mathcal{T}_t$. The dataset of each task consists of a training set $\mathcal{D}_{train}$ and a test set $\mathcal{D}_{test}$. For transfer learning, a model is trained to solve the source task $\mathcal{T}_s$ first. Then, the weights of the trained model are fine-tuned using $\mathcal{D}^t_{train}$ of the target task $\mathcal{T}_t$ and evaluated on $\mathcal{D}^t_{test}$ of $\mathcal{T}_t$. For example, training a load forecasting model of a target customer can be performed in the following two steps. First, train a model for load forecasting, or any other forecasting that is sufficiently relevant to load forecasting, of another customer or many other customers. Then, fine-tune the model using the target customer's data. Because the training procedure of a deep learning model consists of slowly updating weight parameters, fine-tuning is simply defined as additional weight parameter updates using the target task's data. In the recent context of deep learning, transfer learning generally refers to parameter transfer and fine-tuning. The choice of source tasks depends on the availability of datasets and their relevance to the target task. In general, transfer learning does not guarantee a performance improvement [26], but it is usually helpful when the source tasks are sufficiently similar to the target task and the target task's data size is small.

B. META LEARNING

In the last few years, meta learning has received considerable attention in the field of deep learning. The motivation for meta-learning is to be able to learn and adapt quickly from a few examples as in the case of a human being. Whereas traditional machine learning aims to solve a given task from scratch using a fixed learning algorithm, meta learning is intended to improve the learning algorithm with the experience from multiple learning episodes, which covers a distribution of related but different tasks $p(\mathcal{T})$. Hence, meta learning is considered as a mechanism for *learning to learn*.

There are three main approaches to meta-learning of few-shot learning [28]. *Metric* based methods [23], [24] learn a similarity space over embeddings of examples. Estimated distance between the class embedding and the example embedding in the space is used for prediction. *Optimization* based methods [19], [25] adapt the embedding model parameters using few training examples through gradient descent. *Model* based methods [21], [22], [29] learn to store "experience" for estimating parameters of the model for few-shot learning. In this work, we limit our scope to the optimization based methods because they are known to perform well and because they can be flexibly modified depending on the needs of the application. Whereas most of the followings are general explanations of meta learning, some explanations such as adjusting the inner algorithm are specific to the optimization based meta learning.

Meta learning has two components [30]. One component is for learning through an *inner* or *base* learning algorithm with an inner objective such that a particular task can be handled well. The other component is for learning to learn through an *outer* or *meta* learning algorithm with an outer objective such that the meta learner knows how to adjust the inner learning algorithm whenever a new task is given. The existence of an outer objective is the major difference from transfer learning. Whereas meta learning has a wide scope of applications, recent studies have focused on a few applications including few-shot learning [30]. Few-shot learning aims to solve classification or regression problems with only a few labeled examples, and thus fits our need to perform load forecasting with only a limited amount of data from each target individual. As a meta learning setup, N-way K-shot has been heavily studied, and an insightful classification example is shown in Fig. 1. In our work, we focus on adopting the N-way K-shot meta learning setup to the load forecasting because there is a strong analogy between N-way K-shot and 'load forecasting of a customer ($N = 1$) based on the last K days of load usage examples'. An illustrative classification example for a vision application is shown in Fig. 1.

Here we describe meta learning of the N-way K-shot problem. We consider a set of p classes, $\{c_1, c_2, \cdots, c_p\}$, and an unoverlapping set of q classes, $\{c_{p+1}, c_{p+2}, \cdots, c_{p+q}\}$. The p classes are used for training the meta learning model, and the q classes are used for evaluating the model. We denote the dataset for the tasks using the p classes as meta-train dataset, $\mathcal{D}_{meta-train}$, and the dataset for the tasks using the q

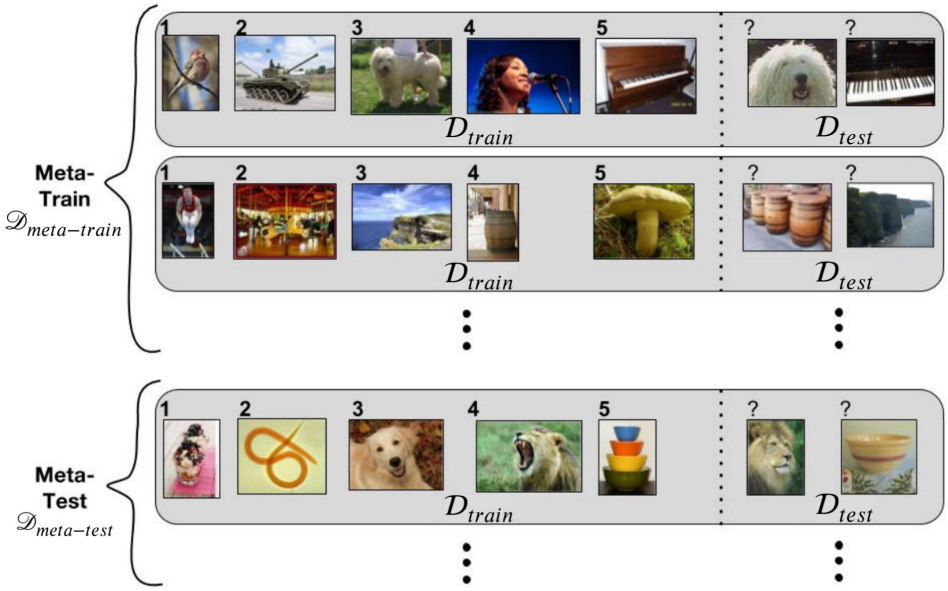

FIGURE 1. Meta learning setup as 5-way 1-shot in [27, Fig. 1]. Each gray box represents a task, where we have one example from each of 5 classes (with each class labeled 1-5) in the training set $\mathcal{D}_{train}$ and two examples in the test set $\mathcal{D}_{test}$. For each task, the goal is to use the five training samples to learn the classes, and then to determine to which class each of the test sample belongs to. The number of training samples in each task is too small (only five) and learning is impossible for traditional learning, but meta-learning can learn general knowledge from many of such tasks that belong to $\mathscr{D}_{meta-train}$ and perform well for the tasks in $\mathscr{D}_{meta-test}$. Each of the meta-training set $\mathscr{D}_{meta-train}$ and meta-test set $\mathscr{D}_{meta-test}$ consists of many tasks, but typically the classes used in the two sets are disjoint in the benchmark problems.

classes as meta-test dataset, $\mathscr{D}_{meta-test}$. For meta-training, a task's training set $\mathcal{D}_{train}$ contains N classes that are sampled from the p classes. For each selected class, K examples are randomly chosen as the training data. The task's test set $\mathcal{D}_{test}$ is formed in a similar way where each test sample is formed by choosing K examples from one of the N classes. For meta-test, all are the same except that the samples are chosen from the q classes.

In meta-training phase, the inner learning algorithm solves tasks from $\mathscr{D}_{meta-train}$ and the outer learning algorithm updates the way inner learning algorithm is adjusted with $\mathcal{D}_{train}$ for each individual task. The outer learning algorithm aims to improve an outer (or meta) objective, and an example of the outer objective can be the average performance of the inner learning algorithm over many individual tasks. In meta-testing phase, the trained outer learning algorithm can quickly adapt the inner model to the chosen task from $\mathscr{D}_{meta-test}$ using only the N-way K-shot examples of the task. Through a large number of tasks from $\mathscr{D}_{meta-train}$, the trained model can generalize well for unseen tasks from $\mathscr{D}_{meta-test}$. There is no intersection between the classes from $\mathscr{D}_{meta-train}$ and the classes from $\mathscr{D}_{meta-test}$, but meta learning works because the underlying features are the same for both $\mathscr{D}_{meta-train}$ and $\mathscr{D}_{meta-test}$.

Here, we introduce the Model-Agnostic Meta Learning (MAML) algorithm [19], which is a representative optimization-based method and is easy to apply to the existing deep learning models. MAML formulates meta learning as a bi-level optimization procedure, where inner optimization aims to adapt to a given task, and outer optimization is intended to generalize unseen tasks during meta training. Specifically, consider a parametric model f_θ with parameters θ. In the inner optimization, the parameter vector θ will be updated as θ_j' as a consequence of one or more gradient descent updates on j'th task $\mathcal{T}_j$ that is sampled from $\mathscr{D}_{meta-train}$ (i.e., $\mathcal{T}_j$ is sampled from task distribution $p(\mathcal{T})$); $\theta_j' = \theta - \alpha\nabla_\theta\mathcal{L}_{\mathcal{T}_j}(f_\theta)$. The inner learning rate (step size) α can be fixed or learned. For simple notation, we consider only one gradient update. In the outer optimization, that is, meta optimization, the model parameters θ are updated using the updated model parameters θ_j' across tasks via stochastic gradient descent as follows:

$$\theta \leftarrow \theta - \beta\nabla_\theta \sum_{\mathcal{T}\sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_j}(f_{\theta_j'})$$

where β is the meta learning rate (step size). Algorithm 1 presents the exact algorithm.

C. BASE DEEP LEARNING MODELS FOR ONE-FOR-ALL AND INDIVIDUAL LEARNING

In this section, we describe LSTM, sequence to sequence, and ResNet models that serve as the base deep learning model for implementing one-for-all and individual learning models. Thanks to the flexibility of deep learning models, any of the base models can be used as the backbone structure.

The Recurrent Neural Networks (RNNs) are intended to generalize feedforward neural networks to sequences. Given

Algorithm 1 Model-Agnostic Meta Learning [19]

Reguire: $p(\mathcal{T})$: distribution over tasks
Reguire: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_j \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_j$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_j}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_j' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_j}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T} \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_j}(f_{\theta_j'})$
9: **end while**

a sequence input, general RNNs estimate a sequence output by capturing temporal correlations between previous and current information. However, it is challenging for traditional RNNs to learn long-range temporal dependencies owing to the problems of vanishing or exploding gradients [31]. LSTM [32] is one of the representative RNNs, which are capable of solving these problems. LSTM defines and maintains an internal memory cell state to keep the long-range dependencies.

The Sequence to Sequence (Seq2Seq) learning model [33] is a multilayered LSTM for encoding and decoding. The encoder-decoder approach can map all the necessary information of an input sequence into a fixed-length vector. The Seq2Seq model was designed for machine translation, but the encoder-decoder framework is also well suited for energy load forecasting [34].

Recently, convolutional neural networks have been the key technique in computer vision and other pattern recognition areas. Not surprisingly, stacking layers cause the problem of vanishing and exploding gradients as in LSTM. ResNet [35] has been developed to solve the gradient problem for CNNs. ResNet is intended to learn the input residual by referencing the layer input. Depending on the number of layers L in convolution and pooling operation, the ResNet model is referred to as ResNet-L.

III. METHODOLOGY

In this section, we present the framework of STLF individualization including traditional, transfer, and meta learning. We also explain the statistical, ARIMA, and one-for-all models that are evaluated together for comparison.

We consider a parametric model f_θ as a predictor that maps a past time series $\mathbf{x}$ to a subsequent future time series $\hat{\mathbf{y}}$ as below:

$$f_\theta : \mathbf{x} \rightarrow \hat{\mathbf{y}}$$

We consider a parametric model f_θ as a predictor that maps a past time series $\mathbf{x}$ to a subsequent future time series $\hat{\mathbf{y}}$ as: $f_\theta : \mathbf{x} \rightarrow \hat{\mathbf{y}}$. In our study, $\mathbf{x}$ is the observed load vector of the customer i for the past 5 days and 12 hours and $\hat{\mathbf{y}}$ is the future load vector of the customer during the subsequent 12 hours. An example is shown in Fig. 2. The ground truth of the future time series is denoted as $\mathbf{y}$. To accommodate the meta learning framework, we consider each customer as a class and split all the customers in our dataset into the meta-training group of p customers, $\{c_1, c_2, \cdots, c_p\}$, and the meta-test group of q customers, $\{c_{p+1}, c_{p+2}, \cdots, c_{p+q}\}$. To accommodate the meta learning framework, we consider each customer as a class and split all the customers in our dataset into the meta-training group of p customers, group P, and the meta-test group of q customers, group Q. The first group is named as group P and the second group is named as group Q. Data of each group can be divided according to the time period, as shown in Fig. 3. We refer to the load data of group P during the training period and test period as $\mathcal{D}^P_{train}$ and $\mathcal{D}^P_{test}$, respectively. Also, we call the load data of group Q during training period and test period as $\mathcal{D}^Q_{train}$ and $\mathcal{D}^Q_{test}$, respectively.

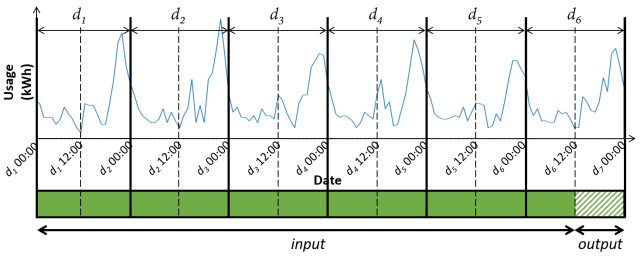

FIGURE 2. An example of input and output. The span of input vector is 5 days and 12 hours. The output is the vector of immediately following 12 hours.

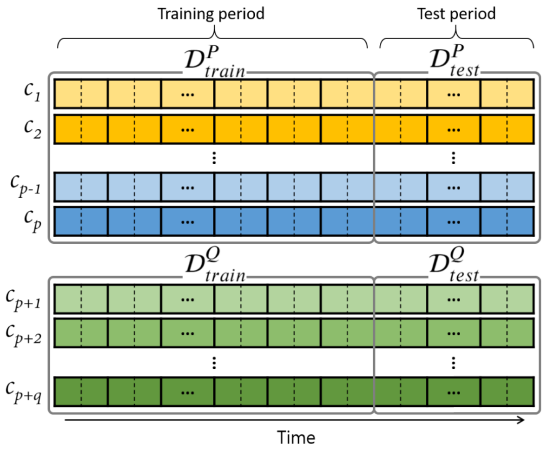

FIGURE 3. Data splits for modeling short-term load forecasting. Each row is a time-series, and it represents electric load data of a customer. For the p customers in group P, $\mathcal{D}^P_{train}$ is the load data for training period and $\mathcal{D}^P_{test}$ is the load data for test period. $\mathcal{D}^Q_{train}$ and $\mathcal{D}^Q_{test}$ are defined in the same way, but using the q customers in group Q instead.

Assuming the data split shown in Fig. 3, the training data for learning the models can be summarized as Table 1. All of the models are evaluated over $\mathcal{D}^Q_{test}$, i.e. over $\mathcal{D}^{c_i}_{test}$ for all $i \in [p+1, \cdots, p+q]$. Assuming the data split shown in Fig. 3, all of the models are evaluated over $\mathcal{D}^Q_{test}$, i.e. over $\mathcal{D}^{c_i}_{test}$ for all $i \in [p+1, \cdots, p+q]$. The overall information on how the

data is used for training and testing are illustrated in Fig. 4. Further details are provided in the following.

TABLE 1. Data used for training customer *i*'s model. Test performance is evaluated over $\mathcal{D}_{test}^{c_i}$.

Model		Data used for training
Statistical methods		$\mathcal{D}_{test}^{c_i}$
ARIMA		$\mathcal{D}_{train}^{c_i} + \mathcal{D}_{test}^{c_i}$
One-for-all learning		$\mathcal{D}_{train}^{P}$
Individual learning	Traditional	$\mathcal{D}_{train}^{c_i}$
	Transfer	$\mathcal{D}_{train}^{P} + \mathcal{D}_{train}^{c_i}$
	Meta	$\mathcal{D}_{train}^{P}$

A. STATISTICAL METHODS, ARIMA, AND ONE-FOR-ALL LEARNING

In our study, some of the statistical methods, such as Avg5, High4of5, Low4of5, and ARIMA are also evaluated for comparison. Avg5 estimates the load as the average load of the last five days. High4of5 selects the top four days out of the last five days, and calculates the average load of the four days. Low4of5 is calculated by the average load value of the bottom four days out of the last five days. These methods are commonly used in industry [36] as the default choice, because they are extremely easy to implement and because they are relatively accurate. Except for ARIMA, there is no need to train a model and only the customer i's own input $\mathbf{x}$ is used for predicting the output $\hat{\mathbf{y}}$. In our experiments, this means the past five days of data is used for the prediction. ARIMA, on the other hand, utilizes all of the historical data of customer i. Therefore, $\mathcal{D}_{train}^{Q}$ is used in addition to $\mathcal{D}_{test}^{Q}$ for training ARIMA.

One-for-all learning is the most conventional way of applying machine learning algorithms to STLF. In one-for-all learning, the models are trained using the data of all p customers in $\mathcal{D}_{train}^{P}$ and evaluated over the q customers in $\mathcal{D}_{test}^{Q}$. Note that $\mathcal{D}_{test}^{P}$ and $\mathcal{D}_{train}^{Q}$ are not used for training.

B. INDIVIDUAL LEARNING

The details of individual models are provided here.

1) TRADITIONAL LEARNING

As shown in Fig. 4, only the historical data of customer i is used for training traditional individual model. The training for customer i is performed using $\mathcal{D}_{train}^{c_i}$, and the evaluation is performed using $\mathcal{D}_{test}^{c_i}$. This model is simple to understand and easy to implement, but it suffers from the small size of $\mathcal{D}_{train}^{c_i}$ and lack of general understanding over all the customers.

2) TRANSFER LEARNING

As shown in Fig. 4, transfer learning first learns from the entire population in group P using $\mathcal{D}_{train}^{P}$, and then adapts to the data of customer i using $\mathcal{D}_{train}^{c_i}$. The first step is performed as a regular training and the second step is performed as a fine-tuning of the model parameters. Evaluation of customer i's model is performed using $\mathcal{D}_{test}^{c_i}$ as in the traditional learning, and the overall performance is found by repeating over the q customers in group Q. Note that transfer learning is different from the one-for-all model only because of the presence of the fine-tuning step.

3) META LEARNING

Our framework of meta learning is shown in Fig. 4. STLF is a regression problem, and we apply 1-way 5-shot learning for predicting customer i's load. In this setup, each task is defined as the following. First, only one customer is involved in each task because N is equal to one. For the task's training data $\mathcal{D}_{train}$, user i's load data of the past five days are provided. In there, each day of train data is divided into the *before noon part* (shown in solid color in the meta learning segment of Fig. 4) and *after noon part* (shown in striped color), and thus allowing the relationship modeling between the before and after parts of each day. In each day, each part of the data consists of 12-hours of consecutive load measurements. Then, the load data of the sixth day is used as the task's test data $\mathcal{D}_{test}$. In the test, the algorithm's goal is to look into the before noon part of the sixth day and to predict the load value vector of the after noon part. For the task defined as above, the meta learning model is trained using the tasks in $\mathscr{D}_{meta-train}$ where the tasks are defined using the dataset $\mathcal{D}_{train}^{P}$. The test of meta learning model is performed using the tasks in $\mathscr{D}_{meta-test}$ where the tasks are defined using the dataset $\mathcal{D}_{test}^{Q}$.

IV. EXPERIMENT

The dataset, metrics, and implementation details are explained here.

A. DATASETS

We performed evaluation over a residential and a non-residential load dataset. The residential dataset is from the Korean demand response pilot program between January 2017 and September 2017 that is explained in [36]. We excluded load data of weekends and replaced the data of holidays and demand response event days, for which people were requested to reduce their load, with the load data of the same day of the previous week. The data contained load data of 3,543 customers. Each set of customer data contained hourly electric loads. The data are aggregated to 1-hour resolution and contained load data of 3,543 customers. We randomly chose 2,633 customers for group P and 910 customers for group Q. Among 2,633 customers in group P, the data of 526 randomly chosen customers were used as the validation set. Additionally, we set the training period from January to July and the test period from August to September because the high temperature and the use of air conditioning caused a high load variance in the summer for the residential data [36] and we wanted to evaluate the modeling effects for the challenging period.

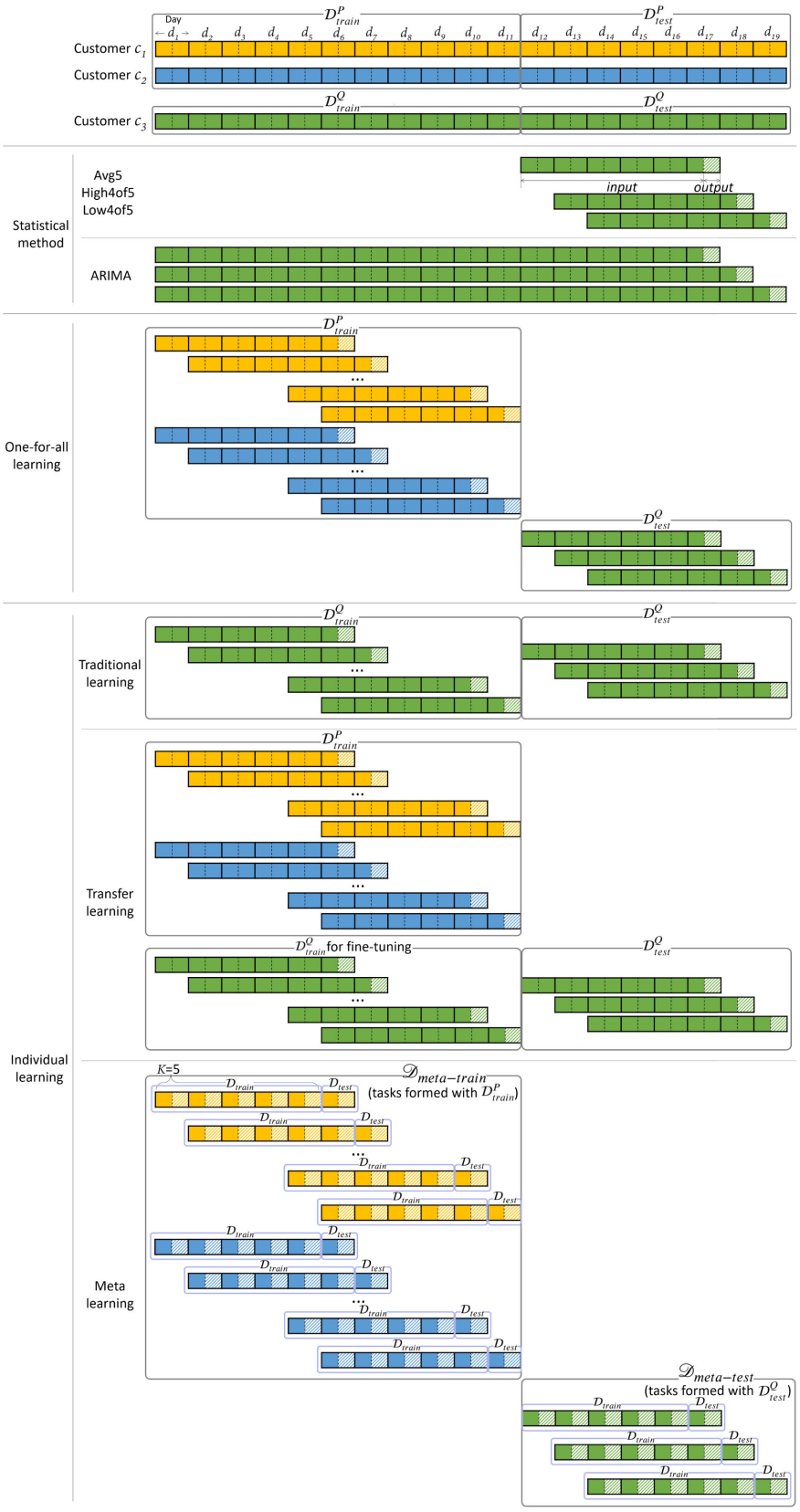

FIGURE 4. The learning models evaluated in this work and the data used for training and testing the models. In this simplified illustration, $i = 1, 2$ are the customers in group P (shown in yellow and blue colors) and $i = 3$ is the target customer in group Q (shown in green color) whose electric load needs to be predicted. In other words, $p = 2$ and $q = 1$.

The non-residential dataset is a substation electric load dataset that is one of the public datasets in the UCI repository [37]. The dataset covers the electric loads of 370 substations in Portugal from 2011 to 2014 with a 15-min sampling period. We aggregated the data as hourly and excluded weekend load data as we did for the residential data. We aggregated the data as hourly as we did for the residential data. Among the 370, we used 334 substations whose data was available between August 2013 and September 2014. The numbers of substations for group P and group Q were 265 and 69, respectively. Randomly chosen 64 substations among the 265 substations were used as the validation set. We trained the models using the data during training period from August 2013 to July 2014, i.e., exactly one year, and the models were evaluated using the data during test period from August to September 2014. Summary statistics of residential and non-residential datasets are shown in Table 2.

TABLE 2. Statistics of hourly load data in kWh.

Dataset	Mean	Median	Std.	Min.	Max.
Residential	0.3707	0.3058	0.2825	0.0075	8.6626
Non-residential	2718.34	543.86	14602.19	0.00	616400.00

B. METRICS

The error metrics used in this study are Root Mean Squared Error (RMSE) and Symmetric Mean Absolute Percentage Error (SMAPE), which are typical metrics used to evaluate STLF models. RMSE and SMAPE are calculated as follows:

$$\text{RMSE} = \sqrt{\frac{1}{C \times D \times M} \sum_{c=1}^{C} \sum_{d=1}^{D} \sum_{m=1}^{M} (\mathbf{y}_{c,d,m} - \hat{\mathbf{y}}_{c,d,m})^2} \tag{1}$$

$$\text{SMAPE} = \frac{100}{C \times D \times M} \sum_{c=1}^{C} \sum_{d=1}^{D} \sum_{m=1}^{M} \frac{|\mathbf{y}_{c,d,m} - \hat{\mathbf{y}}_{c,d,m}|}{(|\mathbf{y}_{c,d,m}| + |\hat{\mathbf{y}}_{c,d,m}|)/2} \tag{2}$$

where $\mathbf{y}_{c,d,m}$ is the m'th element of the ground truth load vector of day d for the customer c during the test period and $\hat{\mathbf{y}}_{c,d,m}$ is the same for the predicted load vector. In our experiments, $C = q$ because the test group (group Q) has q customers and $M = 12$ because we predict the later 12 hours of the sixth day's hourly load values. D varied over the experiments.

C. IMPLEMENTATION DETAILS

In this section, we describe the implementation details of our experiments. For all the models, each individual's electric load measurements were normalized with the average value of its input vector. This was necessary to handle the extensive variations in dynamic range over the customers.

1) STATISTICAL METHODS, ARIMA, AND ONE-FOR-ALL LEARNING

To compare the individual learning models with the existing methods, we used three statistical models (Avg5, High4of5, and Low4of5), ARIMA, Extreme Gradient Boosting (XGB), and four deep learning models (MLP, LSTM, Seq2Seq, ResNet/LSTM). XGB was included because it is one of the most popular and well performing models among the traditional machine learning algorithms. For the training of ARIMA, we used all available days before each forecasting day. The order of AR and MA for ARIMA was optimized for each individual with the maximum order of three for both autoregressive and moving average models. For XGB, we set the total tree number as 1,000, the depth of each individual tree as four, and the learning rate as in [38]. The deep learning models were used for one-for-all and all three individual learning models, and the details of the deep learning models are provided in the following subsections. The details of the deep learning models for one-for-all learning are provided in the following subsections. We used `python` for our experiment. Auto ARIMA in the pmdarima library and XGBRegressor in the XGBoost library were used for ARIMA and XGB. Keras and Tensorflow libraries were used for the deep learning models.

2) TRADITIONAL LEARNING

We evaluated deep learning models including MLP, LSTM [32], Seq2Seq [33], and ResNet [35] of 12 layers with LSTM (ResNet/LSTM). We use four fully connected layers for the MLP model with 256, 128, 64, and 64 units, respectively. The LSTM model comprises seven cells with 300 hidden nodes and a fully connected layer. Seq2Seq model has three LSTM cells with 100 hidden nodes for both the encoder and decoder. We also use the same architecture of ResNet-12 with LSTM as in [14], which consists of twelve-layer ResNet and the following three cells with 300 hidden nodes. All models were trained and evaluated for each customer.

Deep learning models were trained for 150 epochs. Those models were optimized using Adam optimizer with a learning rate of 0.001. Batch size was 128, and activation function was ReLU. We did not apply an activation function on the output layer, the cell output of the last LSTM cell, or any outputs of LSTM cells in ResNet/LSTM as in the typical regression models. These hyperparameters were set the same as the hyperparameters of [14] except for the number of epochs. Their settings were found to be suitable for the proposed work.

3) TRANSFER LEARNING

The deep learning models and their parameters described in IV-C2 were also used for transfer learning. For each model, the number of fine-tuned layers was adjusted according to the validation performance. In the residential data, the last hidden layer and output layer for MLP, the output layer for LSTM, the last three hidden layers and the output layer for Seq2Seq,

and the output layer for ResNet/LSTM were fine-tuned. In the non-residential data, the output layer was fine-tuned for MLP, LSTM, and Seq2Seq, and all layers were fine-tuned for ResNet/LSTM. The number of epochs for fine-tuning was fixed to 10.

4) META LEARNING

For meta learning, only MLP was tried as the base model. We have used MAML that is explained in II-B as the meta learning algorithm, and the other deep learning models such as LSTM, Seq2Seq, and ResNet/LSTM required too much memory for running the training. Meta learning rate β was 0.001 and inner learning rate α was 0.005. For meta learning, we have used MAML as the meta learning algorithm, and only MLP was tried as the base model due to lack of memory for running the training. We have used MAML as the meta learning algorithm, and the other deep learning models such as LSTM, Seq2Seq, and ResNet/LSTM required too much memory for running the training. We have used Adam as the optimizer and have set $\alpha = 0.005$, $\beta = 0.001$. We used ten gradient updates for adjusting the inner algorithm of meta learning and batch size was 128. The number of maximum iterations was 80,000 where early stopping was used.

5) HYPERPARAMETER OPTIMIZATION (HPO)

Deep learning networks are highly sensitive to the choice of tuning parameters, commonly known as hyperparameters in machine learning. Whereas the basic assessments were performed using the algorithms' default hyperparameter values, we desired to observe the extent of improvement that is possible through hyperparameter optimization (HPO). For the most important hyperparameter, the learning rate, we have used diversified Bayesian optimization [39] as the choice of HPO algorithm and assessed the extra improvements. The range of learning rate for one-for-all and transfer learning was $[10^{-6.0}, 10^{-0.2}]$. The ranges of meta learning rate and inner learning rate were both $[10^{-6.0}, 10^{-0.1}]$. The number of updates for meta learning were from $[1, 20]$. For the fine-tuning of transfer learning, the learning rate was kept the same as the learning rate of the base model learning. The number of layers and the number of epochs for fine-tuning were adjusted according to the validation performance.

V. RESULTS

In this section, the experiment results of residential and non-residential datasets are provided. Also, an additional analysis on non-residential dataset is provided where we evaluate the main algorithms' sensitivity to the choice of test period. Important parameters in our experiments are summarized in Table 3.

A. RESIDENTIAL DATASET

For the residential dataset, we evaluated the RMSE and SMAPE performances of nineteen models including statistical models and ARIMA in Table 4, one-for-all and individual models in Table 5. The results of further improvement with HPO are also shown in Table 5.

TABLE 3. The number of customers (substations) for training (p) and for test (q). N and K values used for N-way K-shot of meta learning are also shown.

Dataset	p	q	N	K
Residential dataset	2633	910	1	5
Non-residential dataset	265	69		

TABLE 4. Performance of statistical methods and ARIMA on the residential dataset.

Learning Method		RMSE(kWh)	SMAPE(%)
Statistical methods and ARIMA	Avg5	**0.2806**	33.77
	High4of5	0.2982	35.86
	Low4of5	0.2818	**31.89**
	ARIMA	0.3249	39.04

The three statistical models in Table 4 are very simple, but Avg5 and Low4of5 offer outstanding RMSE and SMAPE performances, respectively. ARIMA, however, performed very poorly despite of using the data from a much longer time period.

One-for-all learning models in Table 5 have shown a large variance in performance depending on which model is used. The best performing model turns out to be XGB that is the only traditional machine learning model.

The evaluations of the ten individual learning models without HPO are shown in Table 5. The first thing to notice is the low performance of the traditional learning models. For the residential dataset, each customer's data spans seven months but still the traditional version of individual learning models did not perform very well. For the transfer learning, MLP and Resnet/LSTM showed good performance but LSTM and Seq2Seq exhibited not so good performance. Meta learning was evaluated for MLP only, and the performance was reasonably good. Overall, transfer learning applied over MLP provided the best performance among the ten individual learning models.

Overall, transfer learning over MLP showed the best performance among the nineteen models. Meta learning over MLP also performed well, and it showed the second best outcome in terms of RMSE.

It is also worth comparing the four MLP models. If we look at the MLP row of Table 5, the following facts can be observed. The one-for-all learning in the first column performs fairly well. The traditional learning, however, ends up with a significant loss in performance presumably because of the smaller size of training data. Both transfer learning and meta learning offer a superior performance even though the improvement over one-for-all is relatively modest. The limited improvement can be due to a few different reasons. First possibility is the characteristics of the residential dataset. Individualization can provide an extra benefit only if there is a sufficient level of diversity among the customers. If not,

TABLE 5. Performance of one-for-all and individual learning methods on the residential dataset.

Model	One-for-all learning		Individual learning					
			Traditional learning		Transfer learning		Meta learning	
	RMSE(kWh)	SMAPE(%)	RMSE(kWh)	SMAPE(%)	RMSE(kWh)	SMAPE(%)	RMSE(kWh)	SMAPE(%)
XGB	**0.2612**	**32.04**	0.2819	32.55				
MLP	0.2614 (0.2610*)	32.23 (32.33*)	0.3063	38.10	**0.2594** (0.2591*)	**31.59** (31.63*)	0.2605 (**0.2586***)	32.07 (**31.31***)
LSTM	0.2951	38.26	0.2804	34.23	0.2817	35.59		
Seq2Seq	0.3091	36.39	0.3099	36.92	0.3075	35.38		
ResNet/LSTM	0.2632	32.74	0.2940	36.58	0.2616	31.70		

* Performance with HPO.

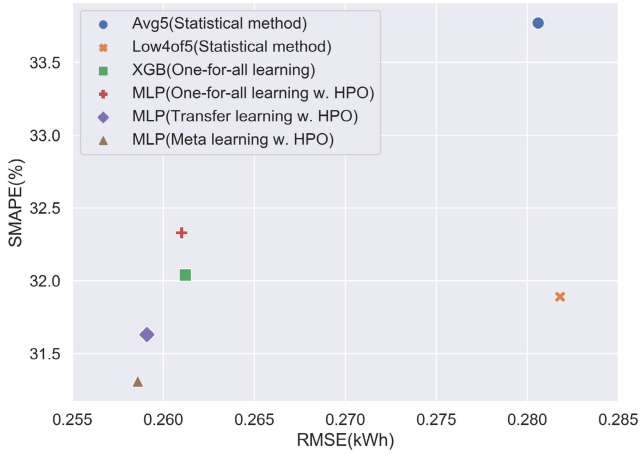

FIGURE 5. RMSE and SMAPE of the most representative algorithms on residential dataset.

the improvement through individualization can be limited. As we will see later in the non-residential results, the gain can be larger for other datasets especially when we consider the seasonal variation. Second possibility is the model training, and we explain the extra improvements that can be gained with hyperparameter optimization in the following.

As mentioned in IV-C5, deep learning models can often have a significant room for improvement. Deep learning models can often have a significant room for improvement. Here, we have considered only the MLP models for HPO, where we have excluded the traditional model because of its low performance in Table 5. For the HPO, we optimized the parameters listed in Section IV-C5 with RMSE as the objective, and the final results are presented together in Table 5 shown inside parenthesis with asterisk mark. If we consider Avg5 as the RMSE baseline and Low4of5 as the SMAPE baseline, transfer learning over MLP decreased RMSE and SMAPE by 7.66% and 0.78%, respectively. Meta learning over MLP outperformed transfer learning, and it decreased RMSE and SMAPE by 7.84% and 1.82%, respectively.

In Fig. 5, a visual summary is shown for the most representative algorithms. Individualization through meta learning achieves the best performance for both RMSE and SMAPE. To double check that the performance difference is statistically significant, paired t-test was performed. For all the pair-wise comparisons, $p < .001$ was obtained confirming the soundness of our result.

TABLE 6. Performance of statistical methods and ARIMA on the non-residential (UCI) dataset.

Learning method		RMSE(kWh)	SMAPE(%)
Statistical methods and ARIMA	Avg5	1464.77	7.17
	High4of5	1607.66	7.63
	Low4of5	**1425.97**	**7.04**
	ARIMA	2451.69	16.70

B. NON-RESIDENTIAL DATASET

We have performed the same experiments using the UCI's non-residential dataset. The results are presented in Table 6, Table 7.

As in the residential case, Low4of5 works very well, as shown in Table 6, and we use Low4of5 as the baseline for both RMSE and SMAPE. Because the non-residential dataset is for substations where loads of numerous residential or industrial customers are aggregated for each substation, the RMSE values are orders of magnitude larger than the results in Table 4.

One-for-all learning models in Table 7 shows a large variance in performance as in the residential case. In particular, the Seq2Seq performed very poorly not only for one-for-all but also for individual learning models. This indicates that Seq2Seq couldn't be trained well for the non-residential dataset. LSTM also performed poorly, but it performed relatively better for the individual learning models. In particular, the poor performance of Seq2Seq indicates that Seq2Seq couldn't be trained well for the non-residential dataset. LSTM also performed poorly, but it performed relatively better for the individual learning models.

The evaluations of the ten individual learning models without HPO are shown in Table 7. This time, the traditional learning over MLP performs well where it works better than Low4of5. However, transfer learning and meta learning over MLP performs even better.

Overall, meta learning over MLP achieved the best performance in terms of RMSE, and transfer learning over MLP achieved the best performance in terms of SMAPE. As in the residential case, the four MLP models can be compared to derive similar analysis.

The best performance results after HPO are presented together in Table 5 shown inside parenthesis with asterisk mark. Compared to the baseline of Low4of5, transfer learning

TABLE 7. Performance of one-for-all and individual learning methods on the non-residential (UCI) dataset.

Model	One-for-all learning		Individual learning					
			Traditional learning		Transfer learning		Meta learning	
	RMSE(kWh)	SMAPE(%)	RMSE(kWh)	SMAPE(%)	RMSE(kWh)	SMAPE(%)	RMSE(kWh)	SMAPE(%)
XGB	1417.97	**6.45**	1664.90	7.96				
MLP	**1315.85** (1253.71*)	6.70 (6.29*)	1675.96	7.50	1299.77 (1240.27*)	**6.28** (**6.17***)	**1231.38** (**1209.72***)	6.31 (6.35*)
LSTM	2570.93	16.31	1686.24	9.04	1734.16	8.92		
Seq2Seq	4006.85	22.36	4013.70	21.91	4056.15	22.16		
ResNet/LSTM	1403.64	7.36	2783.82	24.12	1414.95	7.49		

* Performance with HPO.

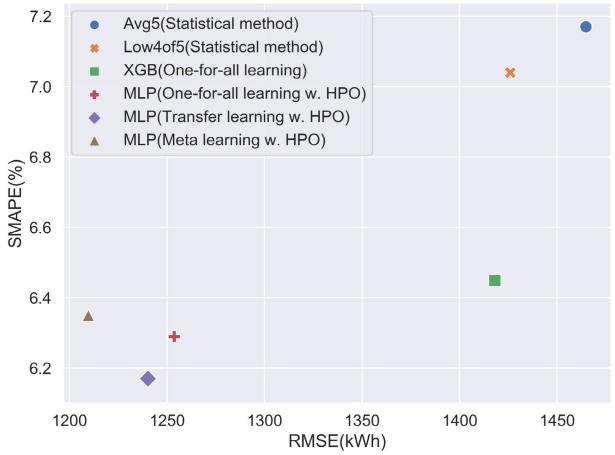

FIGURE 6. RMSE and SMAPE of the most representative algorithms on non-residential (UCI) dataset.

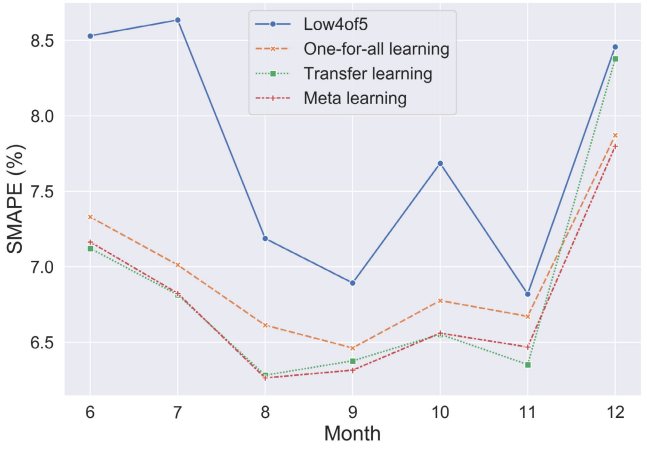

FIGURE 7. SMAPE performance for choosing one month of test period between June and December. Low4of5 and the three individual learning models over MLP are plotted.

over MLP decreased RMSE and SMAPE by 13.02% and 12.36%, respectively. Meta learning over MLP decreased RMSE and SMAPE by 15.17% and 9.8%, respectively.

In Fig. 6, a visual summary is shown for the most representative algorithms. Individualization through meta learning achieves the best performance for RMSE and individualization through transfer learning achieves the best for SMAPE. As in the residential dataset, we performed paired t-test and confirmed that the difference in prediction load is statistically significant ($p < .001$).

C. SENSITIVITY TO TEST PERIOD

The span of residential dataset was between January 2017 and September 2017. Because we wanted to separate the training period and test period as shown in Fig. 3, we have chosen the last two months, August and September, as the test period. For the non-residential dataset, the span was much longer and we were able to include one full cycle of seasons, i.e. one year, into the training period followed by August and September as the test period. We intentionally made the test months to be the same for both residential and non-residential such that there is no difference in terms of seasonal effects.

For the residential dataset, transfer learning and meta learning might have suffered from not having one full year of seasons in the training data. Whereas the test period was August and September, the training data did not contain any information from the previous year's August and September. Therefore, there was no way for the algorithms to learn the load usage characteristics that are unique to the two months. This problem could have been exacerbated for the individual learning models that pay more attention to the specific patterns of individuals. Furthermore, meta learning might have been affected even more because it is supposed to build an individualized predictor based on many possible rules that it has seen in the meta-training dataset. Non-residential dataset's span was long enough that it was possible to alter the test period while still using the previous one year as the training period. Therefore, we have repeated the experiments with one month of test period where the test period ranged between June and December. The results are furnished in Fig. 7. Note that we have chosen only the main four algorithms, Low4of5 and the three individual learning models over MLP, for plotting. Clearly, Low4of5 is the worst algorithm, whereas transfer learning and meta learning are the best algorithms. Whereas we have chosen August and September as the test period in subsection V-B, the performance improvements over Low4of5 are much larger in June, July, and October. Compared to the one-for-all learning, both transfer learning and meta learning generally worked better with one exception of December. In December, the performance of transfer learning is degraded while the performance of meta learning still continues to outstand. It turns out that

December has extended holidays when electric load usage patterns can dramatically change. For the change, transfer learning failed to make proper adjustments and it became even worse than one-for-all learning. Meta learning, however, was able to better handle the change because it uses the input of five training days as the example patterns of what to expect on the sixth day. Overall, the average SMAPE improvement of meta learning over Low4of5 was 12.25% for the seven months shown in Fig. 7.

D. COLD-START PROBLEM

For the existing energy infrastructures, it can be typical to have a sufficient amount of past data. For the emerging sectors, however, only a limited amount of past data might be available - a customer might have just moved in, a new IoT meter might have been just installed, the existing sensors might have been just upgraded to collect samples at a higher frequency (e.g. upgrade from per hour data to per second data), etc. In such cases, some of the customers might suffer from lack of data, and this problem is called the cold-start problem. Formally speaking, cold-start refers to the problem where computer-based information system is not able to draw any inferences for users or items because of the lack of gathered data [40]. There have been a few studies to alleviate the impact of cold-start in STLF. For instance, parallel forecasting models using generative adversarial networks were suggested in [41] and transfer learning models were considered in [4], [42].

For a given target customer, transfer learning requires the customer's past data for individualization. Meta learning, however, does not require the past data of the target customer as can be seen in Fig. 4. To better understand its ramification, we have performed an experiment where the amount of available past data was controlled, and the results are shown in Table 8. It can be seen that transfer learning's performance can be significantly degraded, especially for non-residential dataset. Meta learning's performance, however, is not affected because it does not use the past data of the target customer.

TABLE 8. Cold-start problem - when only a limited amount of past data is available for a customer, transfer learning's performance can be significantly degraded while meta learning's performance is not affected at all.

Dataset	Fine-tuning period	Transfer learning		Meta learning	
		RMSE	SMAPE	RMSE	SMAPE
Residential dataset	7 months	0.2594	31.59	0.2605	32.07
	1 month	0.2624	32.49		
	2 weeks	0.2630	32.57		
	1 week	0.2645	32.51		
Non-residential dataset	12 months	1279.97	6.30	1231.38	6.31
	1 month	1500.94	6.50		
	2 weeks	1441.43	6.62		
	1 week	1509.66	7.51		

VI. CONCLUSION

Short-term electric load forecasting (STLF) is essential for effectively managing energy systems. A reliable analysis through an accurate STLF can be used to secure stable energy supplies or to meet dynamic demands. In this study, we have investigated the potential of individualized modeling for STLF. The latest deep neural network techniques, transfer learning and meta learning, were adopted to implement individualized STLF models for energy data. The individualized models were evaluated on a residential dataset and a non-residential dataset. The RMSE and SMAPE were evaluated, and the proposed models outperformed the existing statistical methods, ARIMA, one-for-all models, and the traditional individual learning approach. Transfer learning and meta learning have been known for many decades, but perhaps we have finally reached the tipping point of harvesting their practical advantages for many energy forecasting problems. This has been made possible thanks to the rapid progress in the deep learning research. The latest deep neural networks are capable enough for modeling complicated patterns and flexible enough for accommodating advanced concepts such as individualization. It is a perfect time to exploit the latest advances in deep learning. For instance, our meta learning approach can learn a reliable and accurate individualized model after just several days of data collection. This can be an important advantage for operating managers like Distribution System Operators (DSOs), because increasingly more energy is being generated locally and connected newly to distribution networks. Additional efforts should be made to identify the key concepts in energy domain that could not be handled well with the traditional methods but might be handled well with the latest technologies.

REFERENCES

[1] E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Deep learning for estimating building energy consumption," *Sustain. Energy, Grids Netw.*, vol. 6, pp. 91–99, Jun. 2016.
[2] X. Cao, S. Dong, Z. Wu, and Y. Jing, "A data-driven hybrid optimization model for short-term residential load forecasting," in *Proc. IEEE Int. Conf. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Autonomic Secure Comput., Pervas. Intell. Comput.*, Oct. 2015, pp. 283–287.
[3] J.-H. Kim and A. Shcherbakova, "Common failures of demand response," *Energy*, vol. 36, no. 2, pp. 873–880, Feb. 2011.
[4] M. Ribeiro, K. Grolinger, H. F. ElYamany, W. A. Higashino, and M. A. M. Capretz, "Transfer learning with seasonal and trend adjustment for cross-building energy forecasting," *Energy Buildings*, vol. 165, pp. 352–363, Apr. 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] H. Leng, X. Li, J. Zhu, H. Tang, Z. Zhang, and N. Ghadimi, "A new wind power prediction method based on ridgelet transforms, hybrid feature selection and closed-loop forecasting," *Adv. Eng. Informat.*, vol. 36, pp. 20–30, Apr. 2018.
[7] Q. Li, Q. Meng, J. Cai, H. Yoshino, and A. Mochida, "Applying support vector machine to predict hourly cooling load in the building," *Appl. Energy*, vol. 86, no. 10, pp. 2249–2256, Oct. 2009.
[8] S. Fan and L. Chen, "Short-term load forecasting based on an adaptive hybrid method," *IEEE Trans. Power Syst.*, vol. 21, no. 1, pp. 392–401, Feb. 2006.
[9] W.-C. Hong, Y. Dong, C.-Y. Lai, L.-Y. Chen, and S.-Y. Wei, "SVR with hybrid chaotic immune algorithm for seasonal load demand forecasting," *Energies*, vol. 4, no. 6, pp. 960–977, Jun. 2011.
[10] F. H. Al-Qahtani and S. F. Crone, "Multivariate k-nearest neighbour regression for time series data—A novel algorithm for forecasting UK electricity demand," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–8.

[11] Y. Zhang, W. Chen, R. Xu, and J. Black, "A cluster-based method for calculating baselines for residential loads," *IEEE Trans. Smart Grid*, vol. 7, no. 5, pp. 2368–2377, Sep. 2016.
[12] N. Ding, C. Benoit, G. Foggia, Y. Besanger, 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.
[13] M. Cai, M. Pipattanasomporn, and S. Rahman, "Day-ahead building-level load forecasts using deep learning vs. Traditional time-series techniques," *Appl. Energy*, vol. 236, pp. 1078–1088, Feb. 2019.
[14] H. Choi, S. Ryu, and H. Kim, "Short-term load forecasting based on ResNet and LSTM," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Oct. 2018, pp. 1–6.
[15] 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.
[16] A. Tascikaraoglu and B. M. Sanandaji, "Short-term residential electric load forecasting: A compressive spatio-temporal approach," *Energy Buildings*, vol. 111, pp. 380–392, Jan. 2016.
[17] E. Mocanu, P. H. Nguyen, W. L. Kling, and M. Gibescu, "Unsupervised energy prediction in a smart grid context using reinforcement cross-building transfer learning," *Energy Buildings*, vol. 116, pp. 646–655, Mar. 2016.
[18] S. Jialin Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[19] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1126–1135.
[20] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–17.
[21] B. Oreshkin, P. R. López, and A. Lacoste, "Tadam: Task dependent adaptive metric for improved few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 721–731.
[22] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," 2018, *arXiv:1807.05960*. [Online]. Available: http://arxiv.org/abs/1807.05960
[23] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. S. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1199–1208.
[24] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4077–4087.
[25] L.-Y. Gui, Y.-X. Wang, D. Ramanan, and J. M. Moura, "Few-shot human motion prediction via meta-learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 432–450.
[26] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.
[27] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. 5th Int. Conf. Learn. Represent.*, Toulon, France, Apr. 2017, pp. 1–11.
[28] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 403–412.
[29] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," 2017, *arXiv:1707.03141*. [Online]. Available: http://arxiv.org/abs/1707.03141
[30] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," 2020, *arXiv:2004.05439*. [Online]. Available: http://arxiv.org/abs/2004.05439
[31] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[33] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[34] D. L. Marino, K. Amarasinghe, and M. Manic, "Building energy load forecasting using deep neural networks," in *Proc. 42nd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2016, pp. 7046–7051.
[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[36] E. Lee, K. Lee, H. Lee, E. Kim, and W. Rhee, "Defining virtual control group to improve customer baseline load calculation of residential demand response," *Appl. Energy*, vol. 250, pp. 946–958, Sep. 2019.
[37] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[38] C. Fan, F. Xiao, and Y. Zhao, "A short-term building cooling load prediction method using deep learning algorithms," *Appl. Energy*, vol. 195, pp. 222–233, Jun. 2017.
[39] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, "Basic enhancement strategies when using Bayesian optimization for hyperparameter tuning of deep neural networks," *IEEE Access*, vol. 8, pp. 52588–52608, 2020.
[40] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*. Cham, Switzerland: Springer, 2007, pp. 291–324.
[41] C. Tian, C. Li, G. Zhang, and Y. Lv, "Data driven parallel prediction of building energy consumption using generative adversarial nets," *Energy Buildings*, vol. 186, pp. 230–243, Mar. 2019.
[42] A. Hooshmand and R. Sharma, "Energy predictive models with limited data using transfer learning," in *Proc. 10th ACM Int. Conf. Future Energy Syst.*, Jun. 2019, pp. 12–16.

EUNJUNG LEE received the B.S. degree in digital media design and applications from Seoul Women's University, Seoul, South Korea, in 2014. She is currently pursuing the Ph.D. degree in transdisciplinary studies with Seoul National University, Seoul. Her current research interest includes time-series data analysis via deep learning, especially meta-learning.

WONJONG RHEE (Fellow, IEEE) received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 1998 and 2002, respectively. From 2001 to 2003, he was with ArrayComm as a Research Staff. From 2004 to 2013, he was with ASSIA Inc. as a Founding Member and later as an ASSIA Fellow. He is currently an Associate Professor with the Department of Transdisciplinary Studies, Seoul National University. His general research interests include data science and machine learning, where information theory, optimization, and signal processing are often utilized.