AugPlug: An Automated Data Augmentation Model to Enhance Online Building Load Forecasting

Yang Deng
Hong Kong Polytechnic Univ.
Hong Kong
yang2.deng@connect.polyu.hk

Rui Liang
Hong Kong Polytechnic Univ.
Hong Kong
maxwell-rui.liang@connect.polyu.hk

Yaohui Liu
Hong Kong Polytechnic Univ.
Hong Kong
yaohui.liu@connect.polyu.hk

Jiaqi Fan
Hong Kong Polytechnic Univ.
Hong Kong
23040866r@connect.polyu.hk

Dan Wang
Hong Kong Polytechnic Univ.
Hong Kong
dan.wang@polyu.edu.hk

ABSTRACT

Online Building Load Forecasting (BLF) is a scheme that designs a model update strategy to continuously update the deployed ML-based BLF model to adapt to changes in the distribution of data. Many online BLF schemes have recently been developed. However, updates can be ineffective, resulting in a decay in accuracy or even in performance that is worse to compared to that without the update. One primary reason for this is poor preparation of the data used to update the model (namely the *updating set*), since most of the online BLF schemes that have been developed update the ML model using collected historical data, which may not reflect the characteristics of the future distribution of data.

To prepare a suitable updating set for the BLF model update is a challenging and ad hoc exercise. In this paper, we propose to leverage *automated data augmentation (AutoDA)*, a data augmentation (DA) framework based on reinforcement learning, to automatically search for the optimal DA policy to generate synthetic data. We thus develop AugPlug, a data augmentation model to instantiate AutoDA in online BLF and demonstrate how it can generate updating sets. A unique advantage of AugPlug is its plug-and-play compatibility for integration into different online BLF schemes. Comprehensive experiments on four published online BLF schemes, involving hundreds of buildings, show that AugPlug can improve the overall performance of the online BLF by 29.37%.

CCS CONCEPTS

• **Applied computing** → **Engineering**.

KEYWORDS

Building load forecasting, Automated machine learning, Data augmentation, Reinforcement learning

BuildSys '24, November 7–8, 2024, Hangzhou, China

ACM ISBN 979-8-4007-0706-3/24/11
https://doi.org/10.1145/3671127.3698190

1 INTRODUCTION

Buildings are major energy consumers and carbon emitters in modern society. In the US, buildings account for over 40% of total energy usage. To better operate building systems and conserve energy, building load forecasting (BLF) plays an important role in many building applications, such as HVAC control, demand response, and others. With the rapid progression of machine learning (ML) during the past decade, many ML-based BLF models have been deployed [16, 20, 27, 28].

Instead of training a BLF model once and continuing to use it thereafter, the practitioner usually needs to continuously update the model with newly collected data. This process is achieved by *online machine learning* and the data used to perform model updates is referred to as *updating set* [41]. Online ML is commonly used in situations where the data are too large to be processed all at once or where the distribution of data is constantly changing[1]. Almost all of the works on online BLF fall into the latter category and are aimed at keeping the accuracy acceptable even if the data distribution changes. For example, an update strategy based on weekly retraining was adopted to update the SVM-based BLF model during the COVID-19 period, with the aim of adapting to changes in human behavior [38]. In most online BLF schemes, the updating set is the recently collected historical data or a selected subset of it. However, the collected data may not be effective at updating the current operational model because the data stream could be continuously drifting. For example, if the cooling demand continues to increase due to the seasonal factor, the forecasts of the updated model could be inaccurate.

Therefore, to achieve acceptable forecasting accuracy in online BLF, it is important to prepare suitable updating sets that reflect the potential characteristics of the coming data distribution for the model updates. Considering that the collected historical data may not directly involve such characteristics, this raises an essential question that remains to be studied: *Can we generate synthetic data as updating sets to support the online BLF model updates, to mitigate the negative impact of changes in data distribution?*

To answer the above question, we designed AugPlug, a data augmentation model based on the *Automated data augmentation* (AutoDA) paradigm [8, 10]. AutoDA is one type of data augmentation that automatically explores the data augmentation (DA) policy

[1]Namely, *concept drift* in the fields related to data science and machine learning.

to improve the quality of the ML model. AutoDA partly resembles the concept of automated machine learning (AutoML), the aim of which is to reduce manual costs to find the optimal strategy. Typical applications that have integrated AutoDA to generate data include Waymo Driver (self-driving), Google Assistant (speech recognition), and others. AutoDA provides us with concrete steps to follow, as well as perspectives on approaches to the solution to which we can refer. Specifically, the proposed AugPlug model materializes key component of AutoDA to search for a DA policy: that is, the search algorithm and the associated search space. Following typical AutoDA tasks, we introduce a reinforcement learning (RL) search algorithm. We formulate the RL process where a recurrent neural network (RNN) based DA policy controller serves as the RL agent to predict the DA policy. We refer to the applications of AutoDA in other time-series fields to design the DA policy search space and finally obtain four types of time-series transformation operations. Moreover, to generate the updating sets for online BLF model updates in our scenario, AugPlug overcomes two unique challenges: how to represent the temporal dynamics of the observed data stream and how to support update strategies in different types of online BLF schemes. To address these two challenges, we designed a Temporal Convolutional Networks layer and an adaptable data transformation module for the DA policy controller.

To show the effectiveness of our approach, we evaluate the proposed AugPlug model to enhance four published online BLF schemes. The evaluation is conducted in 100+ real-world buildings of different building types. We compare AugPlug with two state-of-the-art baselines, including a time-series generation method and a concept drift adaptation method. Our evaluation indicates that the online BLF schemes enhanced by the AugPlug model can achieve an overall improvement in BLF accuracy of 29.37% as compared to the default schemes. Moreover, AugPlug can help the existing online BLF schemes to reduce the percentage of ineffective update operations by 34.73%.

The contributions of the paper can be summarized as follows:

- We investigate the effectiveness of online updates in existing online BLF schemes and empirically show that, contrary to expectations, a significant proportion of these updates have negative effects during the changes in data distribution in the ML deployment phase.
- To the best of our knowledge, this is the first data augmentation solution for improving the performance of the in-operation BLF in deployment. Specifically, we introduce the AutoDA framework for developing a data augmentation model AugPlug[2], to automatically search for the suitable data augmentation policies to enhance the updating set for the online BLF model update. We show that the AugPlug model can improve the effectiveness of the updates and outperforms other solutions.
- As a data augmentation model, AugPlug is plug-and-play and can easily be integrated into online BLF schemes.

[2]The code is available at https://github.com/Dylan0211/AugPlug.

Table 1: Categorization of the online BLF schemes based on the model update strategies.

Adaptation \ Learning mode	*Retrain*	*Fine-tune*
Periodically	SVM [38],RF [34], ensemble [6]	HMM [1], LSTM [18, 42]
Triggered	RF[32], KNN [39], LSTM [26]	RNN [9, 25], GRU [29], AE [17]

2 MOTIVATION AND POTENTIAL APPROACH

2.1 Background on Online Building Load Forecasting

Online BLF refers to the ML-based BLF model that is designed with a model update strategy to enable the model to adapt itself quickly and capture new revealing patterns. Generally, the model update strategies can be differentiated on two dimensions from the online machine learning perspective [4]: adaptation and the learning mode. The former explains how a change in model is initiated, either based on a trigger such as a data distribution change detector or based on a fixed periodic interval such as three months without any explicit detection of change. The learning mode refers to how the model is updated when an adaptation is required. The model can either be retrained from scratch or updated with the most recently collected data. We observed that the update strategy of the majority of existing online BLF schemes falls into this taxonomy.

Another key to the accuracy of a model is the data leveraged to update the model, i.e., the *updating set*. We observed that the majority of schemes prepare the updating set by directly using historical data. For example, in LSTM [18], the LSTM-based forecasting model is continuously updated using historical samples in which the model has previously performed poorly; A sliding-window buffer that keeps recent data is used to update the NN-based forecasting model in [46]. We list some of these works in Table 1. However, only using collected data as the updating set may not involve characteristics that represent the upcoming data. Thus, the update can be ineffective, i.e., the BLF model can not achieve a good performance after the update.

2.2 Motivation

We re-implemented some of the published online BLF schemes and conducted a measurement study on real buildings to simulate the deployment process. We show that online updates can be ineffective. Then, we overview the potential approach.

An Example of Motivation: Figure 1 illustrates variations in BLF performance. Here we use the above-mentioned LSTM [18], one representative online BLF scheme with a *periodically + fine tune* update strategy, which has also been tested in other studies [12, 22]. We tested it on a real university office building with two years' worth of hourly data. First, the initial ML model was well-trained using data from the first-year[3]. Then, in the second year we simulated the model deployment process for 24-hour ahead of the BLF. Here, we compare two accuracies on different modes: (i) freezing the parameters of the initial model and forecasting, and (ii)

[3]The BLF model meets industry requirements: RMSE < 30%, as defined by ASHRAE.

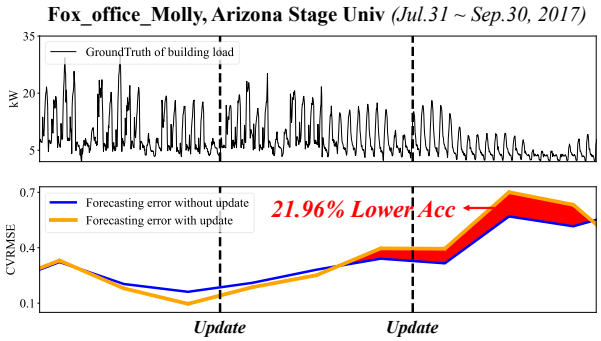

Figure 1: Example of ineffective update in the deployment of scheme [18] (CVRMSE, lower is better).

allowing the model to follow its pre-designed update strategy while forecasting. Figure 1 illustrates an example of a variation in run-time performance during a two-month period. We observe that not only do the updates not always lead to improvements in accuracy, but that even the model with updates does not consistently outperform the frozen model. For instance, during the three-week period from 9 September to 30 September, there was a 21.96% gap in accuracy, indicating that the updates can sometimes be ineffective.

Statistical Analysis: We now determine whether ineffective updates in online BLF are common by testing multiple published online BLF schemes (refer to Table 1) in a large number of buildings.

Datasets and online BLF models: We use a public dataset, Genome [33]. This dataset contains hourly electrical meter data from 1,636 buildings in various countries over a two-year period. For the sake of brevity, we focus on buildings in the USA and leave those in other countries for a future study. We end up with 557 buildings for our analysis. For the BLF schemes, in addition to [18], we re-implemented three more schemes [17, 32, 38] and thus covered all four types of online update strategies in Table 1 (SVM [38] is *retrain + periodically*[4], Random Forest [32] is *retrain + triggered*[5], and Autoencoder [17] is *fine-tune + triggered*[6]. Note that we chose the above four schemes because their design mechanisms are clear and the input features are supported by Genome datasets. Furthermore, the model training and testing settings are consistent with the above motivation example.

Metrics: We employ online A/B testing [5] to quantify the effectiveness of specific update operations. For each update, we compare the accuracy of the updated model with the accuracy of the model had the update not been applied (over the period between this update and the next). We define an update as ineffective if it does not result in an improvement in accuracy.

Results: Table 2 shows the performance of the four tested models across 557 buildings. We compiled statistics on the total number of update operations and the percentage of ineffective updates. We observed that at least 25% of the updates of all of the models were ineffective. In particular, the rate of ineffective updates during the deployment of RF [32] was 32.3% (which conducted the greatest

[4]Update strategy in [38]: weekly retrain based on 30 recent days of data.
[5]Update strategy in [32]: error-based (WAPE) retrain by 20 recent working days of data.
[6]Update strategy in [17]: error-based (CVRMSE) fine-tuned 30 recent days of data.

Online BLF scheme	Num of Update (10^3)	Ratio of Ineffectiveness
SVM [38]	10.7	29.9%
LSTM [18]	6.5	27.3%
RF[32]	15.3	32.3%
AE [17]	11.4	30.7%

Table 2: The proportion of ineffective updates.

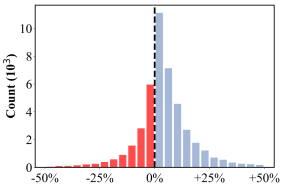

Figure 2: Accuracy improved through updates.

number of update operations). Additionally, we drew a histogram distribution to summarize and visualize the contributions of all updates in Figure 2. The histogram reveals that 11.95% of the updates in fact resulted in a decay in accuracy of more than 10%.

Summary. Our measurements show that the proportion of ineffective updates in online BLF schemes is significant. More importantly, this evaluation was made only in comparison to taking no action at all; thus, we see that there is much room for improvement.

2.3 Potential Approach: Automated Data Augmentation (AutoDA)

The two keys to the accuracy of a model are: the design of the model structure (such as the NN layer, loss function, and hyperparameters), and the data (i.e., the data used to train or update the model should be similar in characteristic to the data in the deployment environment). Modifying the ML model structure in an online scenario is challenging. Moreover, the designs of the BLF models vary greatly, making it unrealistic for the practitioner to calibrate the model case by case. Intuitively, it is more promising to improve the suitability of the updating set to enhance the effectiveness of online BLF model updates. This means generating synthetic data that involves the potential characteristics of the upcoming data to serve as the updating set for each online update operation.

There are several challenges associated with this data generation task. First, while most data generation approaches applied in the building energy field can enhance data diversity and help train ML models to avoid overfitting, they are not suitable for augmenting data when the data distribution changes during the ML model's operational period. Second, it is impractical to directly model all of the dynamics of the building data distribution, especially when the interval between two updates spans a long period. Last, due to the absence of standard metrics to quantify the characteristics of the building data stream, preparing an appropriate update set for each specific update operation requires significant manual effort and expertise. This makes the process inefficient.

We thus leverage *Automated Data Augmentation (AutoDA)*, a subfield of automated machine learning (AutoML) [21], which will provide us with a systematic solution framework. AutoDA is the task of searching for a suitable data augmentation *policy* for a given data set through a learning-based approach. AutoDA has been applied to contribute real products. An example is Waymo Driver, a Google self-driving project. The Google Brain team started applying AutoDA to Waymo in 2019, by leveraging AutoDA to generate traffic data (e.g., pedestrians in various postures and positions) for the image and lidar 3D detection tasks. Meanwhile, the NVIDIA DALI

Table 3: Key Notations and Descriptions.

Symbol	Definition
$M_1, ..., M_j$	the BLF models during the deployment process
D^j_{up}	the original updating set used to update M_j
$\mathcal{F}_{up}(\cdot,\cdot)$	The BLF model update function $\mathcal{F}_{up} : D^j_{up}, M_j \rightarrow M_{j+1}$
D^j_{col}	the collected data (with a fixed size) before updating M_j
π_θ	the DA policy controller RNN for predicting τ
τ	the data augmentation policy $\tau = \{O\}$
O	the data transformation operator contains three parameters: t, λ, p
D^j_τ	the updating set augmented by policy τ, $D^j_\tau = \tau(D^j_{up})$

library[7] introduced an AutoDA module from the 2023 version, with three popular AutoDA algorithms [10, 11, 35]. Intrinsically, The searched DA policy expresses the choices and orders of the data transformation operations, e.g., a sequential application of scaling followed by rotation on point clouds of the pedestrians (to simulate possible movements in the Waymo project).

The key aspect of developing AutoDA is specifying the search algorithm used to find the optimal data augmentation policy, as well as the associated search space. Inspired by AutoML, reinforcement learning (RL) is a typical search algorithm in AutoDA, where the RL agent will explore the DA policies through the feedback reward, i.e., the ML model accuracy. We follow this approach, but we believe that further improvements in the results are possible if better algorithms are used. Besides, the search space of the DA policy is affected by the set of candidate augmentation operations and their parameters, such as the data transformation type.

3 AUTOMATED DATA AUGMENTATION MODEL

In this section, we describe the design of the proposed AutoDA model, AugPlug, including the problem statement, the RL-based framework, the RL agent design, and the AugPlug training method.

3.1 Design Overview

Problem statement: Given an online building load forecast scheme, i.e., an ML-based BLF model will keep updating following a pre-designed update strategy (refer to Table 1), as well as the building to deploy the model, our objective is to maximize the forecasting accuracy of the BLF model during each update process by utilizing the generated update set.

We first present the process and the notations in the typical online BLF scenario, which works as follows: An initial ML-based BLF model M will be trained first. When the model is deployed in a building, the model can be updated with the updating set $D_{up} = \{(x_i, y_i)\}_{i=1}^m$ which is extracted from the collected observations D_{col}, where x is the input features (such as weather and historical load), and y is the forecast load value in one or multiple steps. The model update is performed by further training the model with D_{up}. More formally, the BLF model deployed in a specific building can be denoted as a model sequence of $\{M_1, M_2, ...\}$ for the associated time slots, and the accuracy of a model M_j in its time slot is denoted as $\mathcal{ACC}_{val}(M_j)$ (for the sake of simplicity, we omit the notation of the validation data). The updating strategy

[7]The NVIDIA Data Loading Library (DALI) is a well-known GPU-accelerated library for data loading and pre-processing to accelerate deep learning applications.

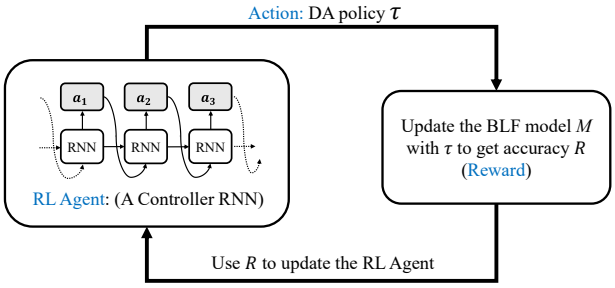

Figure 3: Overview of the AutoDA framework which uses Reinforcement Learning to search for better data augmentation policies. A controller RNN (RL agent) predicts a DA policy τ from the search space. The BLF model is updated to achieve an accuracy R. The reward R will be used with the policy gradient method to update the controller so that it can generate better policies over time.

function $\mathcal{F}_{up}$ specifying the adaptation and the learning mode can be defined as $\mathcal{F}_{up} : D^j_{up}, M_j \rightarrow M_{j+1}$, where M_{j+1} is the updated version[8] of M_j.

We now present AugPlug, a new AutoDA model to benefit the above online model update process. For each update, i.e., given the BLF model M_j, the defined $\mathcal{F}_{up}$, and the historical observations D^j_{col}, then a data augmentation policy τ is applied to transform the original updating set D^j_{up} to a synthetic updating set $D^j_\tau = \tau(D^j_{up})$ that involves potential characteristics of the upcoming future data. The goal of AugPlug is to find the optimal data augmentation policy τ^* that leads to high forecast accuracy of the updated M_{j+1}. Let $M^\tau_{j+1} = \mathcal{F}_{up}(M_j, D^j_\tau)$ be the M_j updated by D^j_τ. The problem can be formulated as follows:

$$\tau^* = \arg\max_\tau \mathcal{ACC}_{val}(M^\tau_{j+1}), \quad j = 1, 2, ... \tag{1}$$

In the following, we first explain how we represent the automated building data augmentation problem in the context of RL (§3.2); Next, we introduce the RL Agent design, the core in the RL framework, to generate actionable DA policies for the online BLF schemes in the various data distributions (§3.3); Finally, we introduce the training and inference algorithm for the RL-based AutoDA model (§3.4).

3.2 Reinforcement Learning Formulation

We represent the RL formulation in our data augmentation scenario (see Figure 3) following the classical AutoDA works that leverages RL as the search algorithm to find the optimal DA policy τ^* in the update process $M_j \rightarrow M_{j+1}$ in a specified building.

At a high level, when the current BLF model M_j calls $\mathcal{F}_{up}$ for updating, the RL agent π_θ (implemented as a controller RNN, which will be shown later) will predict an associated DA policy τ as a list of *actions* $a_{1:T}$ based on the current observed *state* s. Then, the DA policy τ is applied into transform D^j_{up} to a synthetic dataset

[8]For the retrain mode, although a new M_{j+1} is trained from scratch, the structure of the ML model remains the same as that of M_j; thus, we argue that $\mathcal{F}_{up} : D^j_{up}, M_j \rightarrow M_{j+1}$ still applicable for retrain.

Table 4: Time-series transformations and the associated magnitude range.

Type	Description	Magnitudes
Scaling	Multiplies the entire series controlled by λ.	[1,3], [0.3,1]
Jittering	Adds white noise with σ controlled by λ .	[0, 0.1]
Smoothing	Performs low-pass filtering using a average window (with size λ).	(0, 11]
Shifting	Adding λ on the entire series.	[-0.5, 0.5]

D_{τ}^{j}. After $M_{j+1}^{\tau} = \mathcal{F}_{up}(M_j, D_{\tau}^{j})$, we see a *reward* R representing the performance of the online BLF model update. Based on this R, we use reinforcement learning to train the controller RNN. More concretely, to find the optimal DA policy, we ask the controller RNN to maximize its expected reward, represented by $J(\theta)$. The detailed definitions of state, action, and reward are as follows.

$$J(\theta) = E_{P(a_{1:T};\theta)}[R] \tag{2}$$

State: The state is the observed data stream D_{col}^{j} that we can see when updating a BLF model M_j. We define D_{col} contains two classes of observations: i) the collected mechanical and meteorological data, and ii) the historical BLF accuracy records. Both of these two observations are data stream and with the same length. We note that the accuracy records in D_{col} is helpful for exploring τ since it is an intuitive indicator for data distribution change.

the Search Space and Action: Generally, the DA policy τ expresses a list of data transformation operations $\{O\}$ that will be conducted sequentially on the original D_{up}. Each operation O is defined with three parameters: (1) the type of transformation t; (2) the magnitude with which the operation is applied λ; and (3) the probability of applying this operation p.

We follow the existing time-series AutoDA works [36] to design this part, which involves conventional time-series transformation operations such as scaling, jittering, etc., and which are also applied in the building energy scenario to augment data [15]. For example, the operator $O(\text{"}scaling\text{"}, 2.0, 0.8)$ signifies that the transformation operation "scaling" is performed with a probability of 0.8 to scale up the values of the time-series to two times. The DA policy τ can be formulated as Eq.3 and Figure 4 shows a process of applying a τ with two operations on a two-day load time-series. Note that calling probability is primarily employed to introduce stochasticity in policies, which has been demonstrated to be effective [10]. And the identity map (i.e., no augmentation) is also possible with the probability of $(1-p_1)(1-p_2)$.

$$\tau = \{O_n(t_n, \lambda_n, p_n) : n = 1, 2, ..., N\} \tag{3}$$

Taking into consideration the physical characteristics of the time-series data in building applications, we select four transformation operations to suit our scenario (see Table 4). For instance, a jittering operation can simulate malfunctions in building sensors or data collection systems, and scaling (or shifting) can simulate the change in load demands caused by seasons and events. We design the associated magnitude range settings based on data analysis and experiments. Note that, the operator settings are domain-specific and there are works study how to optimize the search space [11]. We put this in future work.

Reward: Our objective is to enhance the accuracy of the BLF model M, while the reward is an improvement in the accuracy in the time slot of M_{j+1}, i.e., from conducting an update on M_j to the next update. Equation 4 shows the definition of the reward.

$$R = \mathcal{ACC}_{val}(M_{j+1}^{\tau}) - \mathcal{ACC}_{val}(M_{j+1}) \tag{4}$$

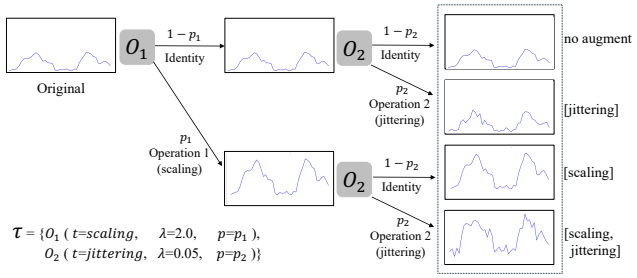

Figure 4: An example of augmented two-day load time-series via a data augmentation policy with two transformation operations. The two operations are applied with the corresponding probabilities.

3.3 RL Agent Design: a Controller RNN

In this section, we first outline the architecture of the DA policy controller. Then, we present two challenges when applying this design to the online BLF scenario, along with the proposed solutions.

We employ a one-layer LSTM with 100 hidden units as the controller RNN for generating DA policies. As shown in Figure 5, the controller outputs a sequence of actions $a_{1:T}$ in an auto-regressive manner, where each a represents the index of a decision regarding a type t, a magnitude λ, or a probability p. That means that the predicted DA policy $\tau = a_1, ...a_T = t_1, \lambda_1, p_1, ..., t_N, \lambda_N, p_N$. At each step in the RNN, the RNN model takes state s, the previous action a_{t-1} and the previous hidden state as inputs. We then apply the softmax layer, utilizing three separate fully connected layers for the three parameters (see Figure 5).

There are two challenges unique to our scenario that prevent direct application of the controller RNN. The first challenge is the representation of the state s, which directs the controller RNN to generate a reliable τ. Generally, The NN-based models design a fully connected layer to encode the input for extracting informative embedding. In our scenario, to identify τ, the temporal dynamics of the observed data stream should be extracted, as it can provide insights into the characteristics of future data. We address this challenge by designing a temporal convolutional network (TCN) embedding layer to represent the state s (Section 3.3.1). The second challenge is the execution of DA policy τ for different types of update strategy $\mathcal{F}_{up}$. Unlike other AutoDA models designed to benefit the specified ML model, in our scenario, the four different types of $\mathcal{F}_{up}$ have varying requirements regarding data size and diversity. We address this challenge by proposing an adaptable data transformation module (Section 3.3.2).

3.3.1 TCN-based embedding layer. we adopt a temporal convolutional network (TCN) [3] to extract the temporal dynamics information from D_{col} because it is superior to other types of neural networks (e.g., MLP or autoencoder) as it exploits convolutional layers with dilated kernels to capture temporal dependencies in samples while maintaining a manageable number of parameters.

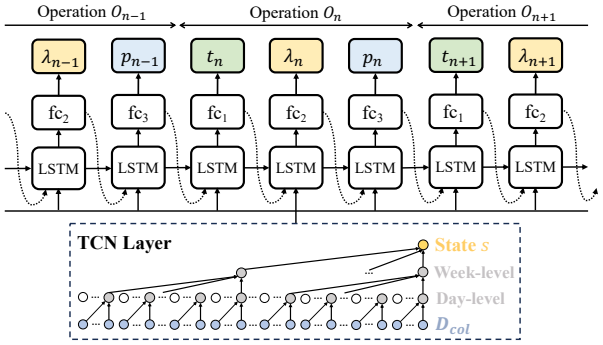

Figure 5: How the controller RNN samples a DA policy. It predicts the type, magnitude, and probability for an operation and repeats.

Generally, the TCN layer is designed into multiple TCN blocks (with different kernel sizes) for the time-series with multi-scale seasonality, which is quite suit for building energy scenarios, i.e., daily and weekly seasonality in terms of occupant behaviors.

As illustrated in the lower part of Figure 5, the designed TCN consists of three blocks with the kernel sizes set to 24, 7, and 4, and the dilation sizes set to 1, 24, and 24*7. This structure is designed to align with the seasonal periods commonly observed in energy time-series (daily and weekly cycles), and corresponds to the data sampling frequency. We consider the output of the last time step as the embedding.

3.3.2 Adaptable data transformation. The online BLF schemes with different update strategies $\mathcal{F}_{up}$ have distinct requirements on data *size* and *diversity* of the generated updating set D_τ, resulting in specific data transformation execution. Intuitively, D_τ for the retrain-based $\mathcal{F}_{up}$ requires larger data size compared to the fine-tune-based one because the former retrains the ML model from scratch. For the adaptation mode, the periodically-based $\mathcal{F}_{up}$ needs more diverse D_τ compared to the triggered-based $\mathcal{F}_{up}$, as the latter usually deal with a relatively deterministic data distribution change.

The data transformed by a specific DA policy τ can be regarded as sampled from the same distribution, we can control the number of transformations conducted (denoted as V) to manage the size of D_{up}. The diversity of D_τ can be increased through the controller RNN predicts multiple different τ with the same state s. We seek the solution from the RL fundamental. There are two types of methods by which the RL agent predicts the action: *argmax*() and *categorical*(), and thus to get a deterministic policy and stochastic policy respectively. Thus, we re-design the RNN output layer as Eq.5, where $h_t = \text{LSTM}([s, a_{t-1}], h_{t-1})$.

$$a_t : \begin{cases} = \text{argmax}(\text{softmax}(\text{fc}_i(h_t))), & \text{// less diversity} \\ \sim \text{Categorical}(\text{softmax}(\text{fc}_i(h_t))), & \text{// greater diversity} \end{cases} \tag{5}$$

, we use U to denote the numbers of the final DA policy τ if τ sampled from *categorical*(). Finally, the synthetic data generated by the τ can be denoted a union set as:

$$D_\tau^j = \cup_{u=1}^{U} \cup_{v=1}^{V} \tau_u(D_{up}^j) \tag{6}$$

Algorithm 1: Training design of AugPlug.

Input: The building dataset $\{\mathcal{D}\}$. The BLF model M and its update strategy $\mathcal{F}_{up}$.
Output: The controller π_θ.

1 Initialize $\mathcal{D}_{train} \leftarrow \emptyset$;
2 **for** $\mathcal{D} \in \{\mathcal{D}\}$ **do**
3 Obtain samples $\{(M_j, D_{col}^j, D_{up}^j)\}$ by deploying M on $\mathcal{D}$;
4 $\mathcal{D}_{train} \leftarrow \mathcal{D}_{train} \cup \{(M_j, D_{col}^j, D_{up}^j)\}$;
5 **for** $i = 1, ..., \#Episodes$ **do**
6 **for** $(M_j, D_{col}^j, D_{up}^j) \in \mathcal{D}_{train}$ **do**
  /* Step 1: prepare updating set */
7 Obtain state s through embedding D_{col}^j with TCN;
8 $\{\tau_u\}_{u=1}^{U} \leftarrow \pi_\theta(s)$;
9 $D_\tau^j \leftarrow \cup_{u=1}^{U} \cup_{v=1}^{V} \tau_u(D_{up}^j)$; // Eq. 6
  /* Step 2: update BLF model */
10 $M_{j+1} \leftarrow \mathcal{F}_{up}(M_j, D_{up}^j)$;
11 $M_{j+1}^\tau \leftarrow \mathcal{F}_{up}(M_j, D_\tau^j)$;
12 $R \leftarrow \mathcal{ACC}_{val}(M_{j+1}^\tau) - \mathcal{ACC}_{val}(M_{j+1})$;
  /* Step 3: update the RL agent */
13 $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{PPO}(\theta)$; // Eq. 7

For the settings of parameters V and U, considering that the most common practice in AutoDA is to conduct transformation one time for a data sample, thus, we use $V = 1$ if $\mathcal{F}_{up}$ requiring a smaller size of D_τ while $V = 2$ for the $\mathcal{F}_{up}$ requiring a larger size. We set the $U = 5$ for the greater diversity referring to an image AutoDA task [10] which also samples five policies to generate data. Besides, we set $V = 2$ for the $\mathcal{F}_{up}$ requiring a larger size and $U = 1$ for less diversity based on our massive experiments.

3.4 AugPlug Training

Given a specific online BLF scheme, the associated AugPlug model will first be trained on multiple building datasets. After that, the AugPlug-enhanced online BLF scheme can be deployed in a target building to forecast building load (will be shown later in §4). In this section, we present the AugPlug training process to maximize the expected reward $J(\theta)$ in Eq.2 for a BLF model M_j. Considering that the reward signal R is non-differentiable, we need to adopt a policy gradient method to iteratively update θ. We employ the Proximal Policy Optimization (PPO) algorithm for our agent training considering its effectiveness and stability. In addition, we choose the clipped surrogate objective to stabilize the training process by limiting the size of the policy change at each step. The loss function is shown as follows.

$$\mathcal{L}_{PPO}(\theta) = -E_{P(a_{1:T};\theta)}[\min(w(\theta)R, \text{clip}(w(\theta), 1-\epsilon, 1+\epsilon)R)],$$

$$w(\theta) = \frac{\pi_\theta(a_{1:T}|s)}{\pi_{\theta_{old}}(a_{1:T}|s)} = \frac{\sum_{t=1}^{T} \log P(a_t|a_{(t-1):1}, s; \theta)}{\sum_{t=1}^{T} \log P(a_t|a_{(t-1):1}, s; \theta_{old})} \tag{7}$$

Next, we outline the flow of our training process (as depicted in Algorithm 1). The training set $\mathcal{D}_{train}$ is first initiated by deploying the studied BLF model on all training buildings to get a sequence of samples where each consists of the model M_j, collected data D_{col}^j, and updating set D_{up}^j within the associated period (line 1-4). For

The file of **inference.py**

```
import AugPlug                                  if __name__ == '__main__':
import numpy, torch, ...                          model = torch.load('BLF_model.pt')
def get_raw_data(): …                             buffer = buffer_mechanism()
def preprocessing(): …                            # BLF model forecasts then updates
def predict(): …                                  while raw_data := get_raw_data():
class buffer_mechanism(): …                         inference_set = preprocessing(raw_data)
# F_up function (periodically + fine-tune)          pred, err = predict(model, inference_set)
def update(model, updating_set):                    buffer.append_data(inference_set)
  for epoch in range(total_iters):                  # original updating set
    for x, y in updating_set:                       updating_set = buffer.output()
      loss = criterion(model(x), y)                 # replace with augmented updating set
      optimizer.zero_grad()                         updating_set = AugPlug(updating_set,
      loss.backward()                                                    raw_data, err)
      optimizer.step()                              update(model, updating_set)
```

Figure 6: An example of integrating AugPlug into the online BLF scheme: LSTM [18].

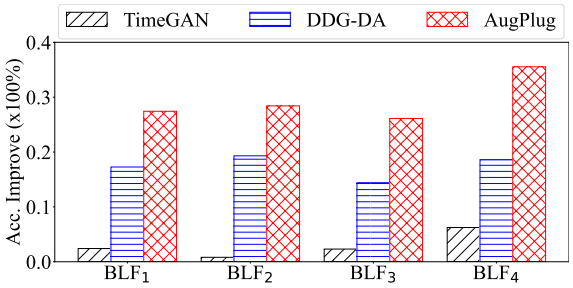

Figure 7: BLF Accuracy improvement through AugPlug and baselines.

each training sample, the state is obtained through embedding D_{col}^{j} with the developed TCN network, based on which the controller π_θ predicts a set of augmentation policies. Then, according to updating strategy of the model, a suitable augmented updating set D_τ^j is generated through the designed data transformation scheme (line 7-9). With the original and augmented updating set, we obtain two versions of updated models, namely M_{j+1} and M_{j+1}^{τ} and their validation accuracy gap (line 10-12), which is considered as the reward. Finally, θ is updated with the PPO loss (line 13).

4 THE ADOPTION OF AUGPLUG

There are two considerations when using the AugPlug model to benefit an online BLF scheme.

The input features of the BLF model: The first consideration is how to augment different categories of features. For an ML-based BLF model, the commonly used input features can be categorized into three types: (1) mechanical features, e.g., history load, power; (2) meteorological features, e.g., outdoor temperature; (3) and time features, e.g., current hour, day of the week [45]. The features designed in all 13 online BLF schemes in Table 1 fall into these three categories. For the first two types, we train separate controller RNNs and perform DA on these features separately if these features are in the BLF model. In particular, we concatenate the input load and output load time-series because they should share the same DA policy [9]. In BLF, time features are typically represented using one-hot encoding. We believe that the temporal dependencies, such as the daily and weekly seasonality of the original samples in D_{up}, should remain unchanged after data augmentation. Additionally, none of the 13 analyzed schemes incorporate month or year indices in their features. Therefore, we directly replicate the time features without any modifications.

Integrating AugPlug into the BLF model: The second consideration is how to integrate the AugPlug model into a developed online BLF scheme to achieve plug-and-play. We still use the LSTM [18] as an example, which is *periodically + fine-tune*. For an ML task (implemented in Python), the code can usually be separated into two parts, i.e., `train.py` and `inference.py`. This work targets the model inference process and there is no revision in `train.py`, where an initial LSTM-based model is trained. The revision in `inference.py` is shown in Figure 6. In this case, besides the basic functions (e.g., data preparation and loading model), `inference.py` specifies the `update()` function, i.e., $\mathcal{F}_{up}$, and the *updating set* is based on its buffer mechanism. As a result, only two additional Python codes are added to replace the original *updating set*. In addition, here we assume that the naming of the .py file and the `update()` function should be regulated; hence the AugPlug training algorithm can smoothly import `update()` (for training the AugPlug model in Algorithm 1).

In summary, there are three steps that a practitioner must follow to use the DA service from AugPlug: (1) Revise the `inference.py` of the online BLF scheme to set a new D_{up} (refer to Figure 6); (2) Train the AugPlug model by Algorithm 1, which imports the user's `update()` from `inference.py`; (3) Run `inference.py` in a target building for load forecasting.

[9]The history load sequence is a key input feature and 13 of the analyzed schemes involve this feature.

5 EVALUATION

In this section, we present the evaluation of the proposed AugPlug model. We first introduce the evaluation methodology. Then, we present the evaluation results, as well as an ablation study to show the key components contributing to the performance of AugPlug.

5.1 Methodology

Online BLF schemes and building datasets: We evaluate the extent to which the accuracy of the online BLF schemes can be improved through the AugPlug model. The tested BLF models and buildings follow the setting in §2.2: SVM [38], LSTM [18], RF [32], and Autoencoder [17]. They correspond to different model updating strategies. We denote them as BLF_1 to BLF_4 in the following experiments. The 557 Genome buildings that we used [33] comprised five main types (i.e., office, education, etc.).

Metrics: To assess the improvement in the BLF accuracy via the proposed AugPlug, we adopt a commonly used metric for evaluating accuracy in the field of load forecasting: the coefficient of varation of the root mean square error (CVRMSE): $\frac{\sqrt{\sum_{i=1}^{n}(y_i-\hat{y})^2/n}}{\sum_{i=1}^{n} y_i/n}$. In addition, we adopt the online A/B testing metrics in §2.2 to assess the proportion of effective updates that were conducted.

Baseline methods: We compare AugPlug with two categories of approaches that also can be leveraged to improve the quality of online BLF model updates: (1) the GAN-based method, the classic data augmentation method that generates more diverse data to update the ML model; (2) the concept drift adaptation method. We leverage one state-of-the-art method that directly forecasts the

Table 5: The CVRMSE (lower is better) of the day ahead load forecasting results on 15 buildings. Comparisons across the default online BLF scheme and the scheme equipped with AugPlug and the baselines.

Online BLF models	**Methods**	*Education*			*Public*			*Assembly*			*Office*			*Lodging*		
		B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	B_{15}
BLF_1	Original	71.48	33.93	32.11	43.53	34.23	49.19	60.27	26.82	43.95	42.97	38.17	44.48	49.75	34.89	40.77
	+ TimeGAN	65.13	37.05	27.75	44.88	35.97	45.31	52.11	27.27	44.89	41.36	34.63	42.85	48.12	38.01	42.66
	+ DDG-DA	53.35	27.43	24.13	39.22	26.81	41.19	46.75	23.53	37.18	38.93	35.07	37.66	43.61	**28.92**	39.34
	+ ***AugPlug***	**41.47**	**22.86**	**23.35**	**33.57**	**23.31**	**36.77**	**33.77**	**18.88**	**32.59**	**28.83**	**25.02**	**31.09**	**31.92**	29.58	**29.43**
BLF_2	Original	54.13	18.53	21.78	23.65	22.01	38.25	46.13	15.03	29.18	28.42	24.86	27.41	36.39	23.24	25.07
	+ TimeGAN	50.24	22.14	19.06	22.31	22.33	36.11	46.94	14.44	26.02	27.55	22.45	26.16	35.32	31.17	26.47
	+ DDG-DA	37.38	16.03	**18.22**	17.35	18.03	31.29	45.29	13.76	27.45	28.54	24.87	23.98	31.22	21.03	25.08
	+ ***AugPlug***	**27.55**	**14.87**	18.71	**16.83**	**15.41**	**28.44**	**32.27**	**13.51**	**23.65**	**25.28**	**18.06**	**23.09**	**19.67**	**16.71**	**20.99**
BLF_3	Original	71.26	35.81	31.26	43.81	36.26	52.41	58.66	33.54	44.59	43.85	42.82	47.31	50.06	38.08	43.47
	+ TimeGAN	56.12	39.12	26.85	40.81	40.54	54.79	51.35	32.14	47.47	40.76	38.44	43.89	48.59	43.78	43.67
	+ DDG-DA	49.34	30.38	**22.74**	40.38	36.42	48.26	43.18	29.93	41.24	35.11	34.51	38.08	47.66	36.86	43.74
	+ ***AugPlug***	**36.22**	**24.27**	24.59	**34.59**	**26.41**	**41.21**	**39.78**	**26.13**	**34.35**	**32.64**	**29.56**	**35.63**	**34.46**	**32.73**	**32.09**
BLF_4	Original	66.52	31.49	22.96	27.58	31.52	26.35	61.62	28.11	32.67	29.42	23.13	24.15	52.52	30.18	42.01
	+ TimeGAN	48.81	30.28	45.89	33.38	29.86	28.92	41.52	22.78	28.36	34.05	26.88	24.06	40.87	30.56	40.78
	+ DDG-DA	44.98	28.26	24.73	26.07	25.89	21.64	37.01	21.72	26.09	30.86	25.14	21.39	30.22	22.17	36.77
	+ ***AugPlug***	**20.12**	**16.39**	**20.31**	**17.21**	**25.32**	**13.44**	**32.38**	**18.16**	**19.58**	**26.28**	**17.54**	**20.23**	**25.12**	**13.17**	**20.33**

future distribution of data. As a result, the following methods are compared:

(1) TimeGAN [44]. This is a state-of-the-art GAN for the time-series field that incorporates temporal dynamics. TimeGAN has been widely used in the energy time-series field, e.g., HVAC control, and heating load prediction. In our scenario, the TimeGAN model is pre-trained on some source buildings and will be fine-tuned with D_{up}. Every time the BLF model needs updates, the TimeGAN model generates synthetic data as the updating set.

(2) DDG-DA [30]. DDG-DA designed a resampling mechanism to sample suitable historical data to approximate the distribution of the incoming data stream. The synthetic data can then be generated from the distribution. This method works well in the public energy dataset [37] for updating the ML forecast model.

Setup of the experiments: To conduct data augmentation, we train the associated AugPlug model for each online BLF scheme. The Genome buildings are split into the training set at 80% and the testing set at 20% (i.e., 445 training buildings and 112 testing buildings, respectively) for the AugPlug and the two baseline methods. The online BLF process is the same as that described in §2: for a test building, the initial BLF model is trained in the first year, and then conducted 24-hour-ahead forecasting as well as the updating in the second year. The proposed AugPlug and the two baseline methods prepare the updating set for each update operation.

5.2 Performance Result

5.2.1 Overall performance. Figure 7 shows the overall improvement in accuracy achieved by AugPlug and the baselines for the BLF schemes over the 112 test buildings. We can see that AugPlug outperforms the baselines on all four BLF schemes. The improvement in accuracy achieved by AugPlug is 27.43%, 28.41%, 26.12%, and 35.53% for the four schemes, respectively. As a comparison, the accuracies of TimeGAN and DDG-DA are much lower, at 2.94% and 17.39% on average. It is worth noting that TimeGAN achieves the least improvement in accuracy among all three methods. This result verifies that conventional DA methods are ineffective in our scenario where data are dynamically changing.

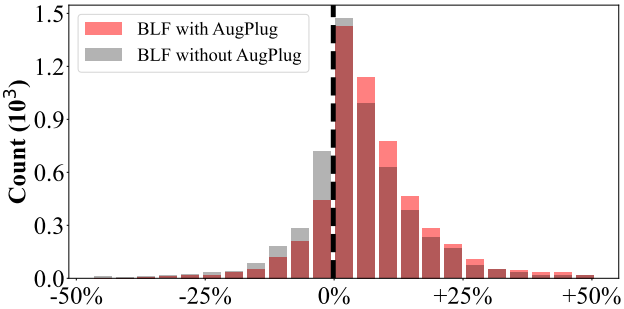

Figure 8: Accuracy improvement (with vs without AugPlug).

Model	**Training time (hour)**	**Inference time (sec)**
BLF_1	70.56	0.43
BLF_2	47.76	0.36
BLF_3	77.04	0.51
BLF_4	59.28	0.42

Table 6: Execution time.

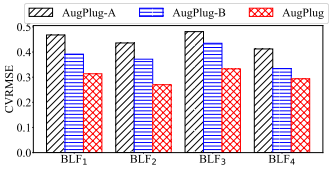

Figure 9: Comparison of AugPlug and the variants.

We drew two histograms in Figure 8 to show the improvement in accuracy of the BLF schemes after updates (following the settings in §2.2). Compared to the histogram of the BLF schemes without AugPlug, the histogram of those equipped with AugPlug shows a noticeable shift to the right, which indicates that the average effectiveness for model updates is enhanced through AugPlug. In detail, AugPlug not only greatly reduces the ratio of ineffective updates (i.e., there was no improvement in accuracy) from 27.2% to 17.7%, but also achieves an overall improvement of 46.8%.

We next study the effectiveness of AugPlug in five different types of buildings. The top three buildings of each type, where the original BLF schemes resulted in the most ineffective updates, are selected.

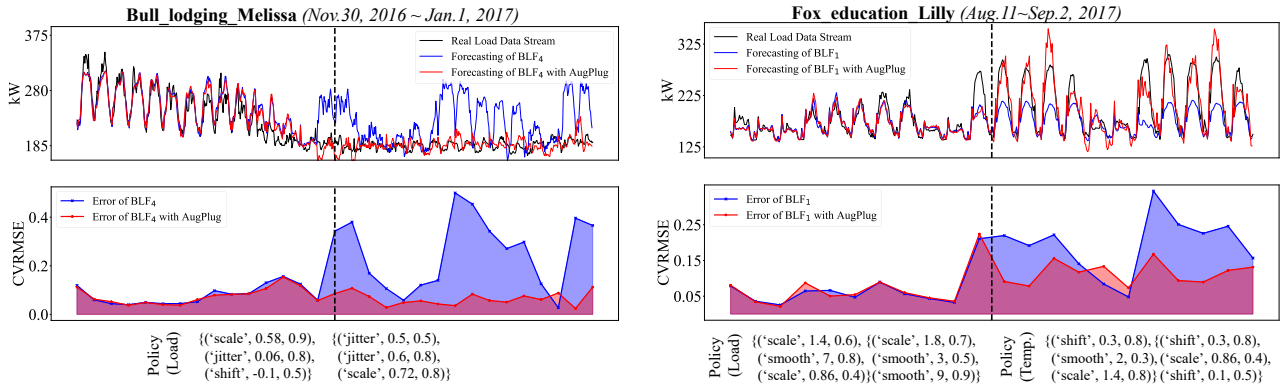

Figure 10: Two examples to show how AugPlug benefits BLF model update. The dashed line indicates a model update. (Top: ground truth and forecasting results. Middle: Error of the BLF under with/without AugPlug. Bottom: the predicted DA policies for the input features from the controller RNN.)

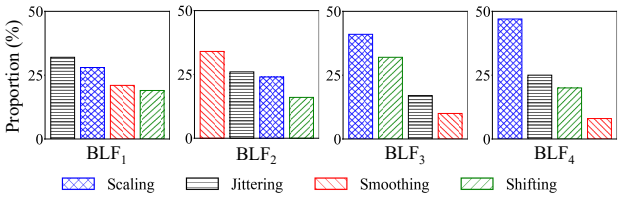

Figure 11: The percentage proportion of data transformation operations for four studied schemes.

Table 5 shows the forecasting accuracy of schemes equipped with AugPlug and baselines on the selected 15 buildings in terms of CVRMSE, where the best-performing method is highlighted in bold font. Overall, AugPlug achieves an average improvement of 30.26% and 19.85% over TimeGAN and DDG-DA, respectively. We also examine that AugPlug nearly outperforms the baselines on all 15 buildings, with improvements of 30.41%, 22.15%, 21.32%, 19.28%, and 29.77% on average for Education, Public, Assembly, Office, and Lodging buildings, respectively. The only exception is that DDG-DA slightly outperforms AugPlug on B_3 and B_{14}. After examining the data in these buildings, we find that the changes in data distribution of the two buildings are regular and steady such that DDG-DA, which is based on resampling historical data for adaptation, quite fits this scenario.

In addition, we study the execution time of the AugPlug model training and inference for the four studied BLF schemes (shown in Table 6), where the training episode is set as 50 for RL convergence. We see that the training process usually takes less than three days and we argue that this is acceptable as the BLF equipped with the trained AugPlug model can be directly deployed in buildings. In addition, the inference time for the BLF schemes equipped with the trained AugPlug model is at most half a second, which is efficient enough for load forecasting under an online setting.

5.2.2 DA Policy analysis. Figure 11 illustrates the proportion in percentages of the transformation operations that were selected for each of the studied BLF schemes. We observe that the proportions of the operations that were adopted differ among the four BLF schemes. This is because different $\mathcal{F}_{up}$ result in diverse data distribution changes from the perspective of in-operation BLF schemes. Specifically, the proportions of the four operations are relatively balanced (between 19% and 32%) in BLF_1, while there are notable differences in the operations conducted in the other three BLF schemes. In particular, in BLF_4, the proportion of scaling is as high as 47%, while that of smoothing is only 8%. We also observe that the rankings of operations among BLF schemes are different, i.e., BLF_3 and BLF_4 conducted the most scaling operations, while BLF_1 and BLF_2 conducted the most jittering and smoothing operations. In detail, we see that scaling and shifting operations are prominent for schemes with triggered-based $\mathcal{F}_{up}$ (i.e., BLF_3 and BLF_4). This is because triggered-based $\mathcal{F}_{up}$ in BLF_3 and BLF_4 are determined by forecasting errors. Therefore, updates are usually performed when the data stream changes in amplitude, which requires scaling and shifting for data transformation. By contrast, for schemes with periodically-based $\mathcal{F}_{up}$ (i.e., BLF_1 and BLF_2), the proportions of the four operations are relatively balanced. This implies that the changes in data distribution encountered by the two schemes are more complex as compared to the other schemes, requiring a more diverse selection of operations.

We then analyze how AugPlug contributes BLF model updates. Figure 10 presents two concrete examples of the deployment of BLF schemes, which perform univariate and multivariate forecasting, respectively. We compare the forecasting results and the accuracy of the BLF scheme equipped with AugPlug against the original scheme. The results show that the predicted DA policies greatly improve the accuracy of BLF schemes after model updates. Specifically, for the first case, the load data value gradually decreases and involves more noise. We observe that the DA policy primarily consists of scaling and jittering operations, which are well-suited to the characteristics of data after update. A similar result is observed in the second case, where AugPlug accurately identified the upcoming increase in load values and reduced instability. Therefore, it generated a DA policy mainly comprised of scaling and smoothing operations to cope with these changes. In addition, since the scheme in the second example takes temperature as an input feature, a separate DA policy is predicted to conduct transformation for temperature data. We

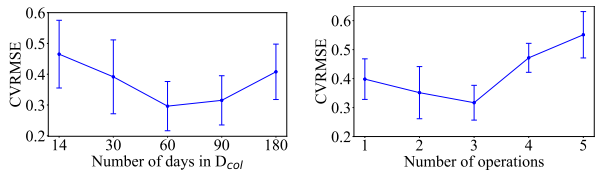

Figure 12: The key parameters setting.

observe that the most frequently selected operation for temperature data is shifting, which differs from the operations commonly chosen for load data. This is because the distributions for the two types of data are different, and so are the way they change. For temperature data, we find that the diurnal temperature difference remains nearly constant while the overall temperature level changes, thus requiring shifting operations to adapt.

5.3 Ablation Study

We implement two breakdown versions of AugPlug to more closely examine the contribution of each component. **AugPlug-A** replaces the TCN-based embedding §3.3.1 with a fully connected layer to embed the input state s. **AugPlug-B** replaces the adaptable data transformation §3.3.2 with a simplified setting where only one DA policy is searched and applied for transformation. The results are displayed in Figure 9, where AugPlug shows a consistently better performance than the other configurations. We see that AugPlug outperforms AugPlug-A and AugPlug-B by 32.65% and 20.69%, respectively, in CVRMSE. More specifically, AugPlug-A has the worst performance of all BLF models, which implies the significance of the specifically designed embedding layer for learning effective representations for energy time-series data. In addition, we find that AugPlug only outperforms AugPlug-B by 12.12% on BLF_4, while achieving a 23.55% improvement on average over the other three schemes. This is reasonable because the update strategy of BLF_4 is *triggered + fine-tune*, which does not require a large and diverse set of updating data. Thus, a simplified transformation scheme can achieve a comparable performance.

In addition, we analyze two key parameter settings in AugPlug: i) the length of collected data stream D_{col} and ii) the number of operations in a DA policy τ. As shown in Figure 12, we observe that the optimal length of D_{col} is two months, outperforming the other configurations by 28.59% on average. As for the number of operations in each policy, the best performance is observed when three consecutive operations are involved, which is 36.82% better than the other settings.

6 RELATED WORK

Our work falls under several categories:

Data augmentation in the building domain. Having been targeted to improve data diversity and the size of training data to improve the accuracy of the ML model, data augmentation (DA) has been a commonly employed technique in various application domains. On the one hand, it can be integrated as a data preprocessing step to enhance the quality of building data analysis results, as improvements are typically expected even given sufficient data. On the other hand, it is extremely useful for enriching building operational data for specific building energy management tasks. In the building energy scenario, a number of studies have exploited both basic DA methods (e.g., random transformation, pattern mixing) [15] and advanced learning-based DA methods to generate synthetic data [13, 19], for example, Conditional GAN was developed to generate load data in the multiple buildings [2]. These DA methods have shown promise in applications such as building occupancy patterns [7], electricity load patterns [15], and Fault Detection and Diagnostics (FDD) [43]. While DA schemes have been used in many building applications, they are usually applied to train an initial model before deployment. In response to these developments, our work takes a significant step forward to benefit the building ML model deployment phase.

Automated data augmentation. Conventional data augmentation applies label-preserving transformations to augment data, however, the specification of data augmentation functions relies heavily on expert knowledge or repeated trials. To address this shortcoming, AutoDA learns the optimal policy to augment the target dataset [8]. AutoAugment takes a reinforcement learning approach to learn the probability and magnitude of applying different transformations to image dataset [10]. PBA [23] and RandAugment [11] study more efficient search methods to reduce the expensive search effort of AutoAugment, for example, RandAugment replaces the augmentation parameters in AutoAugment by uniform values across all transformations and searches for the magnitude and the number of transformations. AutoDA has been applied in such domains as image [10], text [40], and time-series [36] to improve the accuracy of the ML model. Our paper capitalizes on the strengths of AutoDA to forge a novel path in building energy-related tasks. In addition to leveraging RL to search for DA policies, there are also some works based on other search algorithms, such as the evolution algorithm [23] and Bayesian optimization [31]. Recently, in the ML community, a great deal of work has been conducted on improving AutoDA, such as by shortening the policy searching time [31] , by optimizing the search space [11] , and by dynamic policy search [24].

7 CONCLUSION

In this paper, we presented AugPlug, an automated data augmentation model to generate synthetic data as the updating sets for online building load forecasting in the deployment phase. We first showed that the phenomenon of ineffective updates commonly exists in online BLF schemes. Following the automated data augmentation (AutoDA) paradigm, we then developed a reinforcement learning-based AutoDA model to search for the optimal data augmentation policy when the BLF model conducts updates. AugPlug can easily be integrated into existing online BLF schemes. Extensive evaluation with four representative online BLF schemes demonstrates its outstanding performance. We envision AugPlug being integrated with ML training and testing platforms in building scenarios (e.g., AI platform for building HVAC [14]) to achieve energy savings in future smart buildings.

ACKNOWLEDGMENTS

Dan Wang's work is supported by RGC GRF 15200321, 15201322, 15230624, RGC-CRF C5018-20G, ITC ITF-ITS/056/22MX, and PolyU 1-CDKK, G-SAC8.

REFERENCES

[1] V. Álvarez, S. Mazuelas, et al. 2021. Probabilistic load forecasting based on adaptive online learning. *IEEE Transactions on Power Systems* 36, 4 (2021), 3668–3680.

[2] G. Baasch, G. Rousseau, et al. 2021. A Conditional Generative adversarial Network for energy use in multiple buildings using scarce data. *Energy and AI* (2021).

[3] S. Bai, J. Kolter, et al. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[4] Lucas Baier. 2021. Concept Drift Handling in Information Systems: Preserving the Validity of Deployed Machine Learning Models. (2021).

[5] A. A. Benczúr, L. Kocsis, et al. 2018. Online machine learning in big data streams. *arXiv preprint arXiv:1802.05872* (2018).

[6] Yvon Besanger, Quoc Tuan Tran, et al. 2022. Self-updating machine learning system for building load forecasting-method, implementation and case-study on COVID-19 impact. *Sustainable Energy, Grids and Networks* 32 (2022), 100873.

[7] Zhenghua Chen and Chaoyang Jiang. 2018. Building occupancy modeling using generative adversarial network. *Energy and Buildings* 174 (2018), 372–379.

[8] Tsz-Him Cheung and Dit-Yan Yeung. 2023. A Survey of Automated Data Augmentation for Image Classification: Learning to Compose, Mix, and Generate. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[9] G. Chitalia, M. Pipattanasomporn, et al. 2020. Robust short-term electrical load forecasting framework for commercial buildings using deep recurrent neural networks. *Applied Energy* 278 (2020), 115410.

[10] E. Cubuk, B. Zoph, et al. 2019. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 113–123.

[11] E. Cubuk, B. Zoph, et al. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 702–703.

[12] Y. Deng, J. Fan, et al. 2022. Behavior testing of load forecasting models using BuildChecks. In *Proc. of ACM e-Energy '22*.

[13] Y. Deng, R. Liang, et al. 2023. Decomposition-based Data Augmentation for Time-series Building Load Data. In *Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 51–60.

[14] Yang Deng, Donghua Xie, Jingyun Zeng, Rui Liang, Yufei Zhang, Jiaqi Fan, Samson Tai, and Dan Wang. 2024. Towards Deploying ML-based Load Forecasting Models for Building HVAC System: an AI Evaluation Platform. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*. 488–489.

[15] C. Fan, M. Chen, R. Tang, and J. Wang. 2022. A novel deep generative modeling-based data augmentation strategy for improving short-term building energy predictions. In *Building Simulation*.

[16] C. Fan, F. Xiao, and Y. Zhao. 2017. A short-term building cooling load prediction method using deep learning algorithms. *Applied energy* (2017).

[17] Z. Fazlipour, E. Mashhour, and M. Joorabian. 2022. A deep model for short-term load forecasting applying a stacked autoencoder based on LSTM supported by a multi-stage attention mechanism. *Applied Energy* 327 (2022), 120063.

[18] M. Fekri, H. Patel, et al. 2021. Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network. *Applied Energy* 282 (2021), 116177.

[19] M. Fochesato, Fa. Khayatian, et al. 2022. On the use of conditional TimeGAN to enhance the robustness of a reinforcement learning agent in the building domain. In *Proc. of ACM Buildsys '22*.

[20] N. Gugulothu and E. Subramanian. 2019. Load Forecasting in Energy Markets: An Approach Using Sparse Neural Networks. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*. 403–405.

[21] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-based systems* 212 (2021), 106622.

[22] B. Heidrich, N. Ludwig, et al. 2022. Adaptively coping with concept drifts in energy time series forecasting using profiles. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*. 459–470.

[23] D. Ho, E. Liang, et al. 2019. Population based augmentation: Efficient learning of augmentation policy schedules. In *International conference on machine learning*. PMLR, 2731–2741.

[24] Chengkai Hou, Jieyu Zhang, and Tianyi Zhou. 2023. When to learn what: Model-adaptive data augmentation curriculum. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1717–1728.

[25] Rashpinder Kaur Jagait, Mohammad Navid Fekri, Katarina Grolinger, and Syed Mir. 2021. Load forecasting under concept drift: Online ensemble learning with recurrent neural network and ARIMA. *IEEE Access* 9 (2021), 98992–99008.

[26] Y. Ji, G. Geng, et al. 2021. Enhancing model adaptability using concept drift detection for short-term load forecast. In *2021 IEEE/IAS Industrial and Commercial Power System Asia (I&CPS Asia)*. IEEE, 464–469.

[27] A. Jozi, T. Pinto, et al. 2022. Contextual learning for energy forecasting in buildings. *International Journal of Electrical Power & Energy Systems* 136 (2022), 107707.

[28] A. Li, F. Xiao, et al. 2021. Attention-based interpretable neural network for building cooling load prediction. *Applied Energy* (2021).

[29] D. Li, G. Sun, et al. 2022. A short-term electric load forecast method based on improved sequence-to-sequence GRU with adaptive temporal dependence. *International Journal of Electrical Power & Energy Systems* 137 (2022), 107627.

[30] Wendi Li, Xiao Yang, Weiqing Liu, Yingce Xia, and Jiang Bian. 2022. DDG-DA: Data Distribution Generation for Predictable Concept Drift Adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 4092–4100.

[31] S. Lim, I. Kim, et al. 2019. Fast autoaugment. *Advances in Neural Information Processing Systems* 32 (2019).

[32] D. Mariano-Hernández, L. Hernández-Callejo, et al. 2022. Analysis of the integration of drift detection methods in learning algorithms for electrical consumption forecasting in smart buildings. *Sustainability* 14, 10 (2022), 5857.

[33] C. Miller, A. Kathirgamanathan, et al. 2020. The building data genome project 2, energy meter data from the ASHRAE great energy predictor III competition. *Scientific data* (2020).

[34] Jihoon Moon, Sungwoo Park, Seungmin Rho, and Eenjun Hwang. 2022. Robust building energy consumption forecasting using an online learning approach with R ranger. *Journal of Building Engineering* 47 (2022), 103851.

[35] Samuel G Müller and Frank Hutter. 2021. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 774–782.

[36] Liran Nochumsohn and Omri Azencot. 2024. Data Augmentation Policy Search for Long-Term Forecasting. *arXiv preprint arXiv:2405.00319* (2024).

[37] Hugo TC Pedro, David P Larson, and Carlos FM Coimbra. 2019. A comprehensive dataset for the accelerated development and benchmarking of solar forecasting methods. *Journal of Renewable and Sustainable Energy* 11, 3 (2019).

[38] Eric Pla and Mariana Jiménez Martínez. 2023. Dealing with change: Retraining strategies to improve load forecasting in individual households under Covid-19 restrictions. *Energy Reports* 9 (2023), 82–89.

[39] D. Ramos, B. Teixeira, et al. 2020. Use of sensors and analyzers data for load forecasting: A two stage approach. *Sensors* 20, 12 (2020), 3524.

[40] S. Ren, J. Zhang, et al. 2021. Text autoaugment: Learning compositional augmentation policy for text classification. *arXiv preprint arXiv:2109.00523* (2021).

[41] A. Salem, A. Bhattacharya, et al. 2020. {Updates-Leak}: Data set inference and reconstruction attacks in online learning. In *29th USENIX security symposium (USENIX Security 20)*. 1291–1308.

[42] H. Tu, S. Lukic, et al. 2020. An LSTM-Based Online Prediction Method for Building Electric Load During COVID-19. In *Annual Conference of the PHM Society*, Vol. 12. 8–8.

[43] K. Yan, A. Chong, and Y. Mo. 2020. Generative adversarial network for fault detection diagnosis of chillers. *Building and Environment* 172 (2020), 106698.

[44] J. Yoon, D. Jarrett, et al. 2019. Time-series generative adversarial networks. *Advances in neural information processing systems* (2019).

[45] Z. Zheng, Q. Chen, et al. 2018. Data driven chiller sequencing for reducing HVAC electricity consumption in commercial buildings. In *Proceedings of the Ninth International Conference on Future Energy Systems*. 236–248.

[46] Y. Zhou, X. Tian, et al. 2022. Elastic weight consolidation-based adaptive neural networks for dynamic building energy load prediction modeling. *Energy and Buildings* 265 (2022), 112098.