Enhance Load Forecastability: Optimize Data Sampling Policy by Reinforcing User Behaviors

Guangrui Xie[a], Xi Chen[a,*], Yang Weng[b]

[a]*Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, VA 24061, USA*
[b]*School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85281, USA*

Abstract

Load forecasting has long been a key task for reliable power systems planning and operation. Over the recent years, advanced metering infrastructure has proliferated in industry. This has given rise to many load forecasting methods based on frequent measurements of power states obtained by smart meters. Meanwhile, real-world constraints arising in this new setting present both challenges and opportunities to achieve high load forecastability. The bandwidth constraints often imposed on the transmission between data concentrators and utilities are one of them, which limit the amount of data that can be sampled from customers. There lacks a sampling-rate control policy that is self-adaptive to users' load behaviors through online data interaction with the smart grid environment. In this paper, we formulate the bandwidth-constrained sampling-rate control problem as a Markov decision process (MDP) and provide a reinforcement learning (RL)-based algorithm to solve the MDP for an optimal sampling-rate control policy. The resulting policy can be updated in real time to accommodate volatile load behaviors observed in the smart grid. Numerical experiments show that the proposed RL-based algorithm outperforms competing algorithms and delivers superior predictive performance.

*Corresponding author
Email addresses: guanx92@vt.edu (Guangrui Xie), xchen6@vt.edu (Xi Chen), yang.weng@asu.edu (Yang Weng)

Keywords: Forecasting, Sampling-rate control, Reinforcement learning, Markov decision processes, Machine learning

1. Introduction

Load forecasting is an important task for power system planning purposes, as utilities must take actions to keep the supply and demand of electricity in balance based on load forecasts. However, accurate residential load forecasting has become increasingly challenging due to the integration of distributed energy resources (DERs) into smart grids, which has brought much uncertainty into the distribution grids. Over the past decades, advanced metering infrastructure (AMI) has been widely deployed in power grids. AMI is an integrated system of smart meters, communication networks, and data management systems that enables two-way communications between utilities and customers. It permits a number of important functions that were previously impossible or had to be performed manually, such as remote measurement and monitoring of electricity usage and tampering detection (U.S. Department of Energy, 2016).

A plethora of methods have been proposed for load forecasting over the past decades prior to the wide deployment of AMI. Among them, popular methods include, but are not limited to, traditional time series approaches such as autoregressive integrated moving average (e.g., Arora & Taylor, 2018; Rendon-Sanchez & de Menezes, 2019; Nystrup et al., 2020), machine learning approaches such as support vector regression (e.g., Elattar et al., 2010; Jain et al., 2014), neural network (e.g., Kermanshahi, 1998; Ekonomou et al., 2016; Amjady, 2006), and Gaussian process models (e.g., Alamaniotis et al., 2014; Lloyd, 2014). Most of these methods rely on input features such as loads observed in the past, weather conditions, and day types, whose values are accessible in the absence of advanced metering devices.

With the rapid growth of AMI, more and more residential customers are equipped with smart meters. The past decade has seen a growing number of studies dedicated to load forecasting using data collected by smart me-

ters (e.g., Alberg & Last, 2018; Kell et al., 2018; Xie et al., 2018). Modern smart meters have many desirable features that support real-time data interactions, e.g., allowing utilities to adjust the sampling rates remotely (U.S. Department of Energy, 2016). Meanwhile, real-world constraints arise in this new setting which present both challenges and opportunities to achieve high load forecastability.

Fig. 1 illustrates one such real-world constraint and its crucial impact on load forecastability. In smart grids, data sampled from customers in the same neighborhood are often first aggregated at a concentrator; data collected by all concentrators are then transmitted to the data center of a utility for load forecasting (Nimbargi et al., 2016). A majority of utilities rely on wireless technologies (e.g., general packet radio services) for data communication between concentrators and their data centers, and wireless providers (e.g., Verizon, 2017) often impose a daily limit on the bandwidth. Therefore, the amount of data that can be sampled from a neighborhood is limited by the daily transmission bandwidth imposed, which poses a challenge for accurate load forecasting. Given the prohibitive cost to increase the bandwidth (Balachandran et al., 2014; Rahman & Mto, 2013), it is impractical to relax the bandwidth constraint through investment in practice. A question naturally arises in this context: *how to allocate the limited bandwidth to smart meters in a neighborhood to obtain accurate load forecasts overall?* The most commonly adopted industry practice is to sample evenly from each customer subject to the bandwidth constraint (Balachandran et al., 2014). As a sampling policy, however, it is clearly suboptimal: customers often exhibit distinct load behaviors; a uniform sampling policy can result in redundant data being collected from customers with stable load behaviors, while insufficient data being obtained from customers with highly volatile load behaviors. On the other hand, using a carefully designed sampling-rate control policy based on real-time load behaviors can potentially improve data effectiveness and enhance the overall load forecastability.

Recently, researchers have shown an increased interest in adaptive sampling-rate designs for smart meters, but the literature is still relatively scarce.

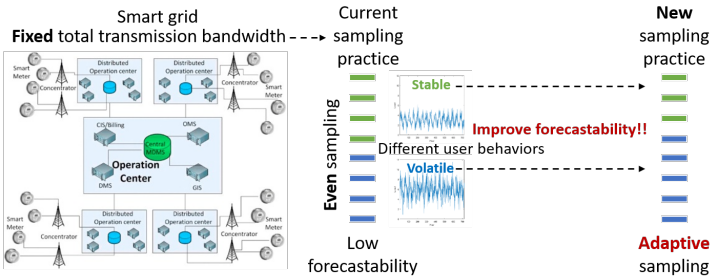


Figure 1: An illustration of the smart-meter sampling-rate control problem, the commonly adopted even sampling practice and an adaptive sampling practice.

Among the first on this topic, Xie et al. (2018) proposed a state-of-the-art, user-behavior-based sampling-rate control algorithm to facilitate load forecasting under a bandwidth constraint. Their sampling policy is obtained via solving an integer program established for a fixed decision horizon and is shown to outperform the even sampling policy. However, as the sampling policy can only be updated periodically with no feedback on the predictive performance being incorporated, it may fail to accommodate highly volatile load behaviors responsively and result in low load forecastability.

To overcome the aforementioned limitations, one can resort to methods capable of supporting online decision making based on data received in real time. One viable choice is online machine learning techniques (e.g., follow-the-leader algorithm, FTL). These methods excel in predictive tasks by learning from continuous streams of data that arrive sequentially. Nevertheless, the goal of online learning methods is to update the corresponding policy parameters such that the regret (e.g., the cumulative predictive error) *in hindsight* can be minimized; and the objective functions typically must possess certain properties (e.g., convexity) for the methods to perform well (Hoi et al., 2018). Markov decision process (MDP), on the other hand, is a rigorous approach to formulate an online decision-making problem whose solution provides an optimal policy that maximizes the expected cumulative future reward (or equivalently, minimizes the expected cumulative future regret), while taking into account the outcomes of all possible future behaviors.

Solving an MDP in many cases, however, can be challenging (e.g., when the transition probabilities are unknown). Reinforcement learning (RL) provides a state-of-the-art solution technique to approximate an optimal policy in these cases and has demonstrated robust performance in practice (Sutton & Barto, 2018).

In this work, we propose to formulate the bandwidth-constrained sampling-rate control problem as an MDP and propose an RL-based algorithm to solve the MDP formulated for an optimal sampling-rate control policy. The MDP is set up to directly maximize the expected future overall load forecastability. The RL-based algorithm can be implemented online to update the sampling-rate control policy adaptively through real-time data interactions. The major contributions of this work include: (1) a novel MDP formulation of the sampling-rate control problem with relatively low action and state space dimensionalities; (2) an RL-based algorithm with provable performance guarantees to solve the MDP formulated for an optimal sampling-rate control policy; and (3) online and offline versions of the RL-based algorithm capable of meeting the needs of different types of customers in the power system.

The rest of the paper is organized as follows. Section 2 reviews a state-of-the-art approach to solving the sampling-rate control problem and reveals the necessity of seeking a new solution. Section 3 presents the formulation of the sampling-rate control problem as an MDP. Section 4 elaborates on the proposed RL-based algorithm for solving the MDP formulated. Section 5 provides numerical experiments on testing the performance of the proposed RL-based sampling algorithm. Finally, Section 6 concludes this work with a summary of its major contributions and a discussion of avenues for future research.

2. Review of an Integer Program-based Approach to the Sampling-Rate Control Problem

The need of an innovation to effectively control smart-meter sampling rates under the bandwidth constraint poses both a challenge and an opportunity for improving the performance of smart grids as mentioned in Section 1.

Xie et al. (2018) were among the first works to propose a smart-meter sampling-rate control policy based on customers' load behaviors. Their sampling policy is obtained by solving an integer program (IP) formulated via a heuristic approach. Specifically, instead of directly maximizing the overall load forecasting accuracy, the IP intends to maximize the weighted total training sample size for all customers subject to the bandwidth constraint. Customers who exhibit highly variable load behaviors are assigned higher weights and those with stable load behaviors are assigned lower weights. The intuition behind this objective function is that the larger training sample size, the higher resulting predictive accuracy. The resulting optimal sampling policy hence adjusts sampling rates for different customers based on their individual load variabilities.

The IP formulation given by Xie et al. (2018), however, suffers from two drawbacks. First, the premise that the objective function relies on is not always true. It is known that some training data points may negatively impact the training process of a prediction model, resulting in poor predictive performance (Fan et al., 2017). Hence, it can be more effective if an objective function can be designed to directly reflect the predictive accuracy achieved on the most recent forecasting periods, serving as the basis for the projection into the next time period. Second, and more importantly, the sampling policy resulting from solving the IP formulated is not responsive to changing load patterns, potentially undermining the predictive accuracy achieved. Specifically, to formulate their IP, a specific decision horizon must be determined first; the decision horizon should be no less than one day due to the daily bandwidth constraint. The IP is updated and solved once for

each decision horizon, and the resulting policy is a deterministic one, with the sampling decisions being held fixed throughout the decision horizon. We refer the interested reader to Xie et al. (2018) for details. In Section 5, we will use the IP-based algorithm as one benchmark for evaluating the proposed approach which is to be detailed in the next two sections.

3. A Markov Decision Process-based Approach to the Sampling-Rate Control Problem

In this section we propose a Markov decision process (MDP)-based approach to the sampling-rate control problem. We first present the prediction model adopted for load forecasting in Section 3.1 and then elaborate on the MDP formulation in Section 3.2. Without loss of generality, we consider hourly prediction for each customer throughout this work. Hence, each stage of the MDP corresponds to each hour in the real world and a decision on whether to sample from each customer must be made at every stage. Other forecast resolutions can be easily adopted without any substantial modification to the MDP formulation.

3.1. The Two-Stage Prediction Approach

To facilitate comparisons with the benchmarking sampling-rate control policy proposed by Xie et al. (2018), in this paper we adopt the same input features and prediction model structure as in Xie et al. (2018). This prediction model falls in the category of two-stage load forecasting models, which typically give predictive performance superior to one-stage models (Bozic et al., 2013). Specifically, assuming the smart grid comprises N customers, the input vector is denoted by

$$\mathbf{x}_t = \left(\theta^t_{i,1}, \theta^t_{i,2}, \ldots, \theta^t_{i,i-1}, \theta^t_{i,i+1}, \theta^t_{i,i+2}, \ldots, \theta^t_{i,N-1}, \theta^t_{i,N}\right)^\top, \tag{1}$$

where $\theta^t_{i,j}$ is the difference in phase angles of customers i and j at hour t. Here, phase angle refers to the lag between the times when a given customer's voltage reaches the peak level and when that happens to the refer-

ence customer in an alternating current system. Phase angles are typically expressed in degrees and are proportional to the time lags which they represent (Grainger & Stevenson, 1994). Load predictions for the next hour are made via the following two-stage approach. The first stage aims at predicting the *hour-ahead* input vector, which contains the values of all pairwise phase angle differences in the next hour. The second stage performs hour-ahead load prediction for a target customer using the predicted input vector obtained by the first stage.

To serve the purpose of first-stage input prediction, we adopt a particular type of recurrent neural network, i.e., gated recurrent unit (GRU) network, which can achieve a higher predictive accuracy for time series forecasting as compared to many other machine learning methods (Cho et al., 2014). GRU has a mechanism of memorizing important temporal patterns while ignoring those unimportant ones seen in the past and has been successfully applied in load forecasting (Zheng et al., 2018). For the second-stage load forecasting, we adopt a Gaussian process (GP) model. GP models have favorable properties such as being highly flexible to capture various features exhibited by the data at hand and capable of quantifying predictive uncertainty (Rasmussen & Williams, 2006).

We note that other input variables that may aid in load forecasting (e.g., weather conditions and day types) can be easily incorporated into the input vector $\mathbf{x}_t$ and used by the aforementioned prediction approach. Moreover, the two-stage prediction approach can easily incorporate other suitable models as the first-stage input prediction model and the second-stage load prediction model.

3.2. The Markov Decision Process Formulation

In this section, we establish the sampling-rate control problem as an MDP, whose solution gives an optimal sampling policy under this formulation.

An MDP is a model for sequential decision making when outcomes are uncertain; it typically consists of decision epochs (or stages), states, actions,

rewards, and transition probabilities. Choosing an action in a state generates a reward and determines the state at the next stage through a transition probability function (which may be known or unknown). Policies or strategies are prescriptions of which action to choose under any eventuality at every future stage. Through solving the MDP formulated, one seeks an optimal policy for choosing an action at each stage so that the total reward accumulated over all stages is maximized (Puterman, 1994).

In our problem setting, the objective function of the MDP is $\max_{\pi \in \Pi} \mathbb{E}_{\pi} \left(\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t) \right)$, where $r_t(s_t, a_t)$ denotes the reward that reflects the predictive accuracy achieved when taking action a_t given state s_t at stage t, Π denotes the set of policies π's that govern actions to take, and $\gamma \in (0, 1)$ denotes the discount factor for future rewards. We elaborate on each component of the MDP next.

3.2.1. Action

At each hour of operation, we must decide which customers to sample phase angles from. Since the reference customer always has a phase angle of zero, there is no need to sample from her. A natural choice to model the action at stage t of the MDP, denoted by a_t, as an $(N-1)$-dimensional vector of binary digits. Each digit corresponds to a distinct customer, with "1" denoting the decision to sample from the corresponding customer at hour t and "0" otherwise. In this case the dimensionality of the action space is $N-1$ and the total number of possible actions at each stage is 2^{N-1}, which is an extremely large action space to explore even for a medium-sized distribution grid.

To avoid a potentially large action space to explore, we adopt a novel approach for modeling action at each stage, which is inspired by the mini-batch idea in Fan et al. (2017). Specifically, we break stage t of the MDP into $N-1$ substages. At each substage, we only consider the action to take for customer i, denoted by a_t^i, for $i = 1, 2, \ldots, N-1$. The dimensionality of the action space at each substage hence reduces to one, with only two possible actions to consider.

3.2.2. State

We take into account the following two aspects when defining the state of the MDP. First and foremost, as a result of the two-stage prediction approach adopted (Section 3.1), the load forecastability ultimately achieved depends heavily on the first-stage hour-ahead input predictive accuracy, which should be captured in the state. In particular, the state at each stage should be able to record the input predictive accuracy achieved for each individual customer, in the same vein as actions being defined at substages corresponding to individual customers (Section 3.2.1). Second, the state should keep track of the remaining budget or bandwidth left for allocation. Therefore, we define the state for customer $i (i = 1, 2, \ldots, N-1)$ at stage t as follows:

$$s_t^i = \left(\bar{e}_t^i, \bar{e}_t^i \left(\sum_{\ell=1}^{N-1} \bar{e}_t^\ell \right)^{-1}, c_t \left(C \right)^{-1} \right), \tag{2}$$

where c_t is the sampling budget remaining at stage t, C is the total daily sampling budget determined by the daily bandwidth constraint, and $\bar{e}_t^i$ denotes the average absolute percentage error for predicting customer i's phase angle up to stage $t-1$, which is defined as

$$\bar{e}_t^i = \frac{1}{t} \sum_{h=1}^{t} \left| \frac{\hat{\theta}_i^{h-1} - \theta_i^{h-1}}{\theta_i^{h-1}} \right|, \ i = 1, 2, \ldots, N-1. \tag{3}$$

Here, θ_i^h denotes the phase angle of customer i observed at hour h and $\hat{\theta}_i^h$ denotes its estimate given by the first-stage input prediction model (recall Section 3.1). The three components of s_t^i respectively account for the input predictive error incurred for customer i, the ratio of customer i's input predictive error to the total input predictive errors incurred for all customers and the percentage of the remaining sampling budget.

At stage t, after taking actions a_t^i's, we need to obtain states s_{t+1}^i's for stage $t+1$. Some difficulty may arise in calculating the component $\bar{e}_{t+1}^i$ in s_{t+1}^i via (3), if some θ_i^t is not observed due to the action a_t^i taken. To proceed,

we can make up the missing phase angle observation by using the first-stage input prediction. Specifically, denote the phase angle vector of the customers at stage t (excluding the reference customer) by $\Theta_t = (\theta_1^t, \theta_2^t, \ldots, \theta_{N-1}^t)^\top$ and its estimate by $\hat{\Theta}_t = (\hat{\theta}_1^t, \hat{\theta}_2^t, \ldots, \hat{\theta}_{N-1}^t)^\top$. For each element in Θ_t, we replace θ_i^t with $\hat{\theta}_i^t$ if the former is not observed; denote the resulting vector by Θ_t^*. Upon updating the first-stage input prediction model with Θ_t^*, we predict Θ_t again and denote the prediction by $\hat{\hat{\Theta}}_t = (\hat{\hat{\theta}}_1^t, \hat{\hat{\theta}}_2^t, \ldots, \hat{\hat{\theta}}_{N-1}^t)^\top$. Then $\bar{e}_{t+1}^i$ can be obtained by replacing $\hat{\theta}_i^t$ and θ_i^t with $\hat{\hat{\theta}}_i^t$ and $\hat{\theta}_i^t$ in (3), respectively.

3.2.3. Reward

As the goal of the MDP is to maximize the overall load forecastability, we define the reward as the load predictive accuracy achieved for all customers. At stage t, we make a prediction for Θ_{t+1} using Θ_t^* and denote it by $\hat{\Theta}_{t+1}$. Then, we obtain the input vector $\mathbf{x}_{t+1}$ based on $\hat{\Theta}_{t+1}$ (recall (1)); $\mathbf{x}_{t+1}$ is subsequently used as the input to the second-stage prediction model for predicting each customer's load at stage $t+1$. In particular, a separate second-stage load prediction model G_i is constructed for performing customer i's load prediction. The reward at stage t of the MDP is defined as

$$r_t = 1 - \frac{\sum_{i=1}^{N} (\hat{P}_i^{t+1} - P_i^{t+1})^2}{\sum_{i=1}^{N} (P_i^{t+1} - \bar{P}^{t+1})^2}, \tag{4}$$

where $\hat{P}_i^{t+1}$ and P_i^{t+1} respectively denote the load prediction obtained and the actual load observed for customer i at hour $t+1$; and $\bar{P}^{t+1} = N^{-1} \sum_{i=1}^{N} P_i^{t+1}$. We see from (4) that the maximum possible reward at each stage is one; and there is a possibility of getting a negative reward. In accordance with the definitions of action and state of the MDP, we assign a unique reward r_t^i to

action a_t^i at substage $i (i = 1, 2, \ldots, N-1)$ as follows:

$$r_t^i = \frac{r_t}{N-2} \left(1 - \bar{e}_t^i \left(\sum_{\ell=1}^{N-1} \bar{e}_t^\ell \right)^{-1} \right), \tag{5}$$

where $\bar{e}_t^i$ is defined in (3). The definition in (5) ensures that the actions corresponding to those customers with lower phase angle estimation errors earn higher rewards and vice versa. Note from (4) and (5) that the sum of the rewards earned at all substages (i.e., corresponding to all customers) equals the total reward earned at stage t, i.e., $\sum_{i=1}^{N-1} r_t^i = r_t$.

4. A Reinforcement Learning-based Solution to the MDP Formulated

One can solve an MDP for an optimal policy via methods such as dynamic programming (DP) and reinforcement learning (RL). While DP works well for solving MDPs with known transition probabilities, RL is more effective when transition probabilities are unknown, as is the case in our problem setting (Sutton & Barto, 2018). In this section, we first briefly introduce a model-free policy-based RL approach, the enhanced REINFORCE method, which serves as the basis of our proposed algorithm to solve the MDP formulated in Section 3. Then, we elaborate on the proposed algorithm in Section 4.2.

4.1. A Policy Gradient Algorithm—REINFORCE

There has been an increasing interest of the power systems community in using RL to solve real-world problems, such as demand response, load control, and electric vehicle fleet charging, to name a few (Lu et al., 2019; Claessens et al., 2018; Ruelens et al., 2017).

RL algorithms typically fall into two categories: value based and policy gradient algorithms; and the latter type tends to converge faster than the former type (Sutton & Barto, 2018). Different from the algorithms that seek

an optimal policy based on value functions (e.g., Q-learning, SARSA), policy gradient algorithms gradually improve the policy by using the gradient of policy parameters. Specifically, the policy is described by a parameterized machine learning model, such as logistic regression or neural network. Such a model takes the state as the input and produces a probability of taking each possible action as the output. Let $\pi_{\boldsymbol{\beta}}$ denote the parameterized policy model with $\boldsymbol{\beta}$ being the d-dimensional parameter vector, r_t as the reward, s_t as the state, and a_t as the action at stage t. A policy gradient algorithm aims at maximizing the expected total reward defined as

$$J(\boldsymbol{\beta}) = \mathbb{E}_{\pi_{\boldsymbol{\beta}}} \left(\sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t) \right) = \sum_{\tau} \pi_{\boldsymbol{\beta}}(\tau) \tilde{r}(\tau),$$

where $\gamma \in (0, 1)$ denotes the discount factor for future rewards, τ is a trajectory that contains a sequence of state-action pairs, i.e., $(s_1, a_1, s_2, a_2, \ldots)$, $\pi_{\boldsymbol{\beta}}(\tau)$ denotes the probability of producing τ given the policy parameter vector $\boldsymbol{\beta}$, and $\tilde{r}(\tau)$ denotes the total discounted reward over all decision horizons under trajectory τ. Seeking an optimal policy is equivalent to finding an optimal parameter vector $\boldsymbol{\beta}^*$ that solves $\max_{\boldsymbol{\beta} \in \mathbb{R}^d} J(\boldsymbol{\beta})$.

Policy gradient algorithms, fittingly, use gradient-based methods to find $\boldsymbol{\beta}^*$. Denote $\nabla J(\boldsymbol{\beta})$ as the gradient of $J(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$. Under standard assumptions on the regularity of the MDP problem and the smoothness of the policy model π_β, one can write $\nabla J(\boldsymbol{\beta})$ in the following form according to the policy gradient theorem (Sutton et al., 2000):

$$\nabla J(\boldsymbol{\beta}) = (1-\gamma)^{-1} \mathbb{E}_{(s,a) \sim \rho_{\boldsymbol{\beta}}(\cdot,\cdot)} \left[\nabla \log \pi_{\boldsymbol{\beta}}(a|s) Q_{\pi_{\boldsymbol{\beta}}}(s, a) \right],$$

where $\rho_{\boldsymbol{\beta}}(s, a) = \rho_{\pi_{\boldsymbol{\beta}}}(s) \pi_{\boldsymbol{\beta}}(a|s)$ denotes the discounted state-action occupancy measure, $\rho_{\pi_{\boldsymbol{\beta}}}(s) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t p(s_t = s | s_0, \pi_{\boldsymbol{\beta}})$ is a probability distribution over the state space $\mathcal{S}$, and $p(s_t = s | s_0, \pi_{\boldsymbol{\beta}})$ denotes the probability that the state at time t equals s given the initial state s_0 and the policy $\pi_{\boldsymbol{\beta}}$. Given an initial state-action pair (s, a), the value of the Q-function gives the expected accumulation of discounted rewards, i.e.,

$Q_{\pi_{\boldsymbol{\beta}}}(s,a) = \mathbb{E}_{\pi_{\boldsymbol{\beta}}}\left(\sum_{t=0}^{\infty}\gamma^t r_t(s_t,a_t)|s_0=s,a_0=a\right)$.

REINFORCE is a classical policy gradient algorithm (Sutton & Barto, 2018), which updates the policy parameter vector $\boldsymbol{\beta}$ via a stochastic gradient ascent approach. The agent learns the policy by interacting with the environment for a large number of episodes, each consisting of many stages. In each episode, the agent starts with some state s_0, takes actions according to the policy parameterized by $\pi_{\boldsymbol{\beta}}$, and observes the reward earned at each stage. At the end of each episode l, $\boldsymbol{\beta}$ is updated based on the trajectory of states, actions, and rewards earned:

$$\boldsymbol{\beta}_{l+1} = \boldsymbol{\beta}_l + \alpha_l \hat{\nabla} J(\boldsymbol{\beta}_{\boldsymbol{l}}), \tag{6}$$

where $\{\alpha_l \in (0,1)\}$ denotes the sequence of step sizes and $\hat{\nabla} J(\boldsymbol{\beta})$ denotes an estimate of $\nabla J(\boldsymbol{\beta})$. One common drawback of classical REINFORCE algorithms (Williams, 1992) is that their resulting $\hat{\nabla} J(\boldsymbol{\beta})$ can be biased, rendering a lack of performance guarantees.

In this work, we adopt an enhanced REINFORCE method which is inspired by Zhang et al. (2020). The method updates the policy parameter vector via (6) using the following gradient estimate:

$$\hat{\nabla} J(\boldsymbol{\beta}) = \frac{1}{1-\gamma}\hat{Q}_{\pi_{\boldsymbol{\beta}}}(s_T,a_T)\nabla \log \pi_{\boldsymbol{\beta}}(a_T|s_T), \tag{7}$$

where T is geometrically distributed with parameter $1-\gamma$, i.e., $T \sim Geo(1-\gamma)$, and $\hat{Q}_{\pi_{\boldsymbol{\beta}}}(s,a)$ denotes the estimated Q-function value given a state-action pair (s,a), specifically,

$$\hat{Q}_{\pi_{\boldsymbol{\beta}}}(s,a) = \sum_{\ell=0}^{T'}\gamma^{\frac{\ell}{2}} r_t(s_\ell,a_\ell)\big|s_0=s,a_0=a, \tag{8}$$

with $T' \sim Geo(1-\gamma^{\frac{1}{2}})$ and T' being independent of T.

The enhanced REINFORCE method has desirable theoretical properties such as producing an unbiased estimate of $\nabla J(\boldsymbol{\beta})$ and the resulting $\boldsymbol{\beta}_l$ con-

verging to a stationary point of $J(\boldsymbol{\beta})$ almost surely. Therefore, convergence to an optimal policy parameter vector $\boldsymbol{\beta}^*$ can be guaranteed. For the sake of brevity, we refer the interested reader to Appendix A for more details.

4.2. The Proposed RL-based Algorithm

In this section, we provide an RL-based algorithm in light of the enhanced REINFORCE method to solve the MDP formulated in Section 3 for an optimal sampling-rate control policy.

With a policy model specified, seeking an optimal policy reduces to seeking an optimal parameter vector $\boldsymbol{\beta}^*$ based on some training dataset. In this work, we model the policy $\pi_\beta(a|s)$ using a multilayer perceptron (MLP), which is a feedforward shallow neural network (NN). It is more computationally efficient than deep NNs thanks to its small scale and has greater flexibility in modeling nonlinearity as compared to non-NN models. Hence, MLP strikes a good balance between computational efficiency and predictive accuracy for problems with low input and output dimensionalities (Hastie et al., 2009). Since the dimensionalities of the space and action spaces of the formulated MDP are not high, an MLP suffices for modeling the sampling policy. At substage i within stage t, the input to the MLP is the state s_t^i and the output is the probability of taking each possible action $a_t^i \in \{0, 1\}$, $i = 1, 2, \ldots, N-1$, $t = 1, 2, \ldots$.

Below we provide two versions of the algorithm, the offline and online versions, respectively suitable when the training dataset is static (i.e., data stay fixed after being recorded) and dynamic (i.e., data are continually updated).

4.2.1. The Offline Version

With a given dataset $\mathcal{D}$ that contains observations from N customers at $\mathcal{T}$ consecutive hours, the offline version of the proposed RL-based algorithm (i.e., Algorithm 1 provided in Appendix B) can be adopted to obtain an optimal sampling-rate control policy. Let L denote the total number of training episodes. We focus on explaining the key steps in Algorithm next.

Steps 1 and 2 of Algorithm 1 respectively initialize the policy model $\pi_{\boldsymbol{\beta}}$ and train separate second-stage load prediction GP models for individual customers. At the beginning of each training episode, Step 4 calculates the initial state for each customer. Step 5 samples T and T' independently from respective geometric distributions for the current training episode. Steps 6 to 24 generate actions based on observed states and calculate the rewards by taking corresponding actions. By the end of the current episode (Steps 25 to 28), the policy parameter vector $\boldsymbol{\beta}$ is updated based on the states, actions, and rewards obtained at all stages within the episode via (6). At the end of the Lth episode (i.e., the last training episode), the policy parameter vector $\boldsymbol{\beta}$ is expected to converge to an optimal parameter vector $\boldsymbol{\beta}^*$ under the MDP formulated in Section 3.2. The resulting policy model $\pi_{\boldsymbol{\beta}^*}(a|s)$ can be used for sampling-rate control in the future.

4.2.2. The Online Version

The online version of the algorithm intends to continually update the sampling-rate control policy based on streaming data. Thanks to the sequential nature of the policy gradient updating scheme, we can update the policy parameter vector $\boldsymbol{\beta}$ on an hourly basis if implemented online instead of at the end of each episode. Specifically, at each hour t, one can update $\boldsymbol{\beta}$ via $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \alpha(1-\gamma)^{-1} r_t^i \cdot \nabla \log \pi_{\boldsymbol{\beta}}(a_t^i|s_t^i)$ for $i = 1, 2, \ldots, N-1$ with a step size $\alpha \in (0, 1)$. Fig. 2 shows a schematic diagram of the online version of the proposed algorithm. Specifically, an initial policy is first obtained upon training offline over a total of L episodes on a given dataset $\mathcal{D}$ via Algorithm 1. Then using the current policy at hour t, sampling decisions and load forecasts can be made for hour $t+1$. As time proceeds to hour $t+1$, the rewards at hour t can be calculated as the data at hour $t+1$ stream in. The policy can be subsequently updated using the newly obtained rewards, states, and actions at hour t. The updated policy can be further applied to determine the actions to take at hour $t+1$. This online version fully ensures that the sampling-rate control policy can be updated and carried out continually, enabling the sampling policy to accommodate highly dynamic

customers' load behaviors.

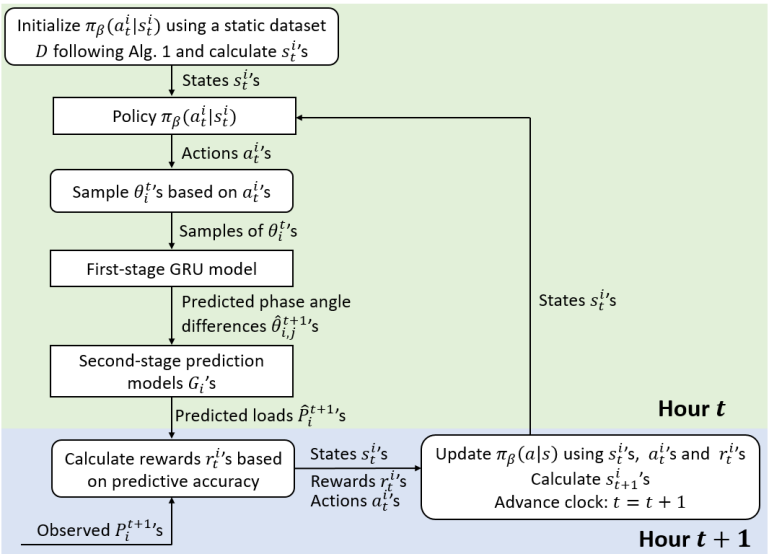


Figure 2: A schematic diagram for the online version of the RL-based sampling algorithm.

5. Numerical Results

In this section, we conduct numerical evaluations of the proposed RL-based algorithm on various test cases.

5.1. Experiment Setup

Data generation. The proposed algorithm is evaluated on four standard IEEE test cases, i.e., 8-bus, 14-bus, 24-bus, and 123-bus test cases, respectively built on real-world data sources. In each test case, a bus refers to an individual customer whose load needs to be predicted over time. To simulate highly uncertain load behaviors caused by DERs in real-life power systems, historical load profiles from PJM (2014) in year 2014 and New York Independent System Operator (2015) in year 2015 are used for simulations. Taking into account the uncertain renewable generation behaviors of DERs, we first pre-process the simulated hourly PV generation data over a year drawn from Renewables.ninja (2017), and then subtract the pre-processed

data from the load data of each customer. To obtain phase angle values, we perform power flow analysis to generate power states hourly over a year using the MATLAB Power System Simulation Package (MATPOWER, Zimmerman & Murillo-Sanchez, 2010) based on the processed load data.

Bandwidth constraint. We assume that without the daily bandwidth constraint, originally it was possible to sample the phase angle θ from all customers every hour; that is, the total number of θ's (excluding a given reference customer) that can be sampled every day was $24(N-1)$, where N denotes the total number of customers in a given test case. Due to the bandwidth constraint imposed, however, the daily total bandwidth is reduced by 1/3; that is, the total number of θ's that can be sampled every day becomes $16(N-1)$.

Algorithm configuration. In each test case, upon obtaining one year's load and phase angle data for each test case, the first 30 days' data are used as the static dataset $\mathcal{D}$ to train the sampling-rate control policy offline via Algorithm 1. Each training episode of the proposed RL-based algorithm has a maximum of $\mathcal{T} = 720$ stages (or hours). The discount factor γ is set to 0.95. The step size is set as $\alpha_l = 1/l$ for $l \geq 1$. The total number of training episodes L is set to 500, to ensure the convergence of the algorithm. Upon completing the offline training through L episodes, the resulting policy is applied to a test set that comprises the remaining 335 days' data, for which the online version of the algorithm is implemented for controlling sampling rates and performing hourly load prediction. Regarding the GRU model used for the first-stage input prediction, the number of hidden units in the GRU cell is set to 5, and the tanh function is selected as the activation function. Regarding the policy model MLP describing $\pi_\beta(a|s)$, the number of hidden layers and the number of hidden units are set to 1 and 10, respectively. All the aforementioned parameters are chosen using a time series cross-validation procedure. Specifically, a set of candidate values for each aforementioned parameter (i.e., the number of hidden units in the GRU cell, the number of hidden layers and the number of hidden units in the MLP model) is adopted to perform predictions on a validation set for each test

case via the rolling-origin-update evaluation of Bergmeir & Benítez (2012). The validation set for each test case contains the first 30 days' data. The measure of predictive error, MAPE, as defined in (9), is calculated across all hours in the validation set for each test case using each combination of parameter values. Finally, the combination giving the lowest average MAPE across all test cases is adopted.

Measure of predictive accuracy. To evaluate the overall predictive accuracy, we consider two performance measures, respectively, the mean absolute percentage error (MAPE) and the mean absolute scaled error (MASE), achieved by hourly predictions for all customers in each test dataset. The MAPE and MASE at each hour $t \in \{1, 2, \ldots, N^*\}$ in a test dataset are respectively defined as

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\hat{P}_i^t - P_i^t}{P_i^t}\right|, \quad \text{MASE} = \frac{\frac{1}{N}\sum_{i=1}^{N}\left|\hat{P}_i^t - P_i^t\right|}{\frac{1}{(N^*-24)N}\sum_{h=25}^{N^*}\sum_{i=1}^{N}\left|P_i^h - P_i^{h-24}\right|}, \quad (9)$$

where P_i^t denotes the load of customer i at hour t actually observed and $\hat{P}_i^t$ is the predicted value, N denotes the total number of customers in a given test case, and N^* denotes the total number of hours to be predicted in each test dataset. As 335 days' data are used for testing, N^* equals $8,040$.

Benchmarking algorithms. We consider three benchmarking algorithms in comparison with the proposed RL-based algorithm (abbreviated to "RL"): (1) the commonly adopted even sampling practice as reviewed in Section 1 (abbreviated to "CP"), (2) the IP-based algorithm as reviewed in Section 2 (abbreviated to "IP") and (3) the follow-the-leader algorithm (abbreviated to "FTL"), a state-of-the-art online machine learning algorithm. For implementing IP, a 48-hour decision horizon is adopted following the suggestion of Xie et al. (2018). FTL can update its policy parameter vector $\boldsymbol{\beta}$ on an hourly basis according to an objective function of minimizing the prediction error accumulated so far. For the reader's convenience, we provide schematic diagrams that illustrate the implementations of IP and FTL respectively in

Fig. C.1 and Fig. C.2 in Appendix C. To ensure the fairness of comparisons, the same load forecasting model (i.e., GRU combined with GP) is applied in conjunction with all four sampling-rate control algorithms.

5.2. Summary of Results

On convergence. Fig. 3 shows that the offline training via Algorithm 1 indeed achieves fast and reliable convergence in all four test cases. Recall that the maximum possible reward earned at each stage is 1 and the maximum possible total reward in one episode given the fixed training dataset $\mathcal{D}$ is 720. In all test cases, we observe from Fig. 3 that the corresponding total reward earned increases rapidly with the number of training episodes performed, and the reward converges within the first 50 episodes to a value near 700. This indicates that RL can successfully solve the MDP formulated for the sampling-rate control problem and obtain an approximated optimal sampling policy numerically.

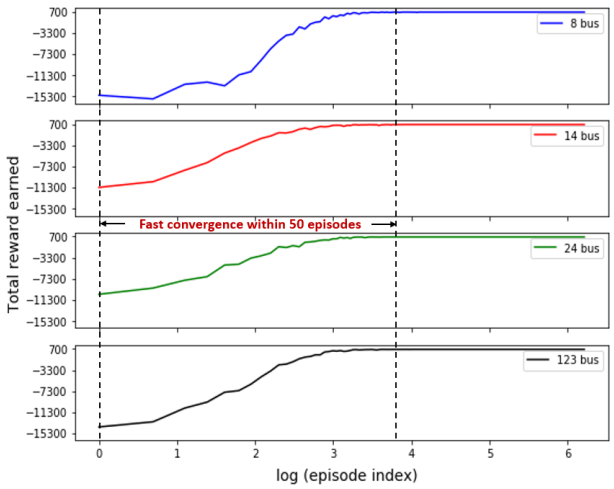


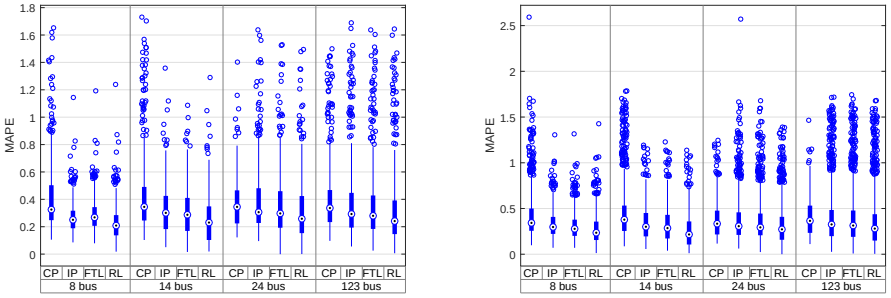
Figure 3: Plots of the total reward earned versus the training episode index (in logarithmic scale) through offline training in each test case.

On predictive performance. Fig. 4 summarizes the MAPEs corresponding to 8,040 hourly load predictions obtained by implementing CP, IP, FTL, and RL in each of the four test cases. As customers typically exhibit distinct

load behaviors on weekdays and weekends, to control for the effects of type of day on the performance of all sampling algorithms under comparison, we present the results obtained for weekdays and weekends separately. From Fig. 4, we see that, for each algorithm considered, there tend to be more outliers in the MAPEs for the weekends than for weekdays; indeed, customers' load behaviors are typically more volatile and hence more challenging to predict on weekends as compared to weekdays. More importantly, Fig. 4 shows that RL dominates the three benchmarking algorithms by producing the lowest MAPEs in all four test cases. IP and FTL's performance is comparable while FTL has a slight edge over IP; and CP ranks last overall. The observation above holds for comparisons with respect to both weekdays and weekends. We further note that the benefit of using an adaptive sampling algorithm (i.e., IP, FTL, and RL) diminishes slightly as the total number of customers increases; this is because as more and more customers are considered for sharing the given bandwidth, the data deficiency issue becomes more challenging to address. Nevertheless, the performance of RL is still robust in this case. Last but not least, similar observations are made from the MASEs obtained. To economize on space, we refer the reader to Fig. C.3 in Appendix C for details.

Table 1 (respectively Table 2) summarizes the medians of the 8,040 MAPEs (resp. MASEs) obtained by each algorithm in all test cases, which further confirms the superiority of RL as observed in Fig. 4 (resp. Fig. C.3). The sign test (Diebold & Mariano, 1995) conducted shows that the medians of the MAPEs (resp. MASEs) of CP, IP, and FTL are significantly higher than that of RL at a confidence level of 95%.

On responsiveness. Fig. 5 (respectively Fig. 6) compares load predictions on two consecutive days for a representative customer with stable load behaviors (resp. volatile load behaviors) in each of the four test cases. We observe from Fig. 5 for customers with stable load behaviors that, in the 123-bus test case, on the first day, IP, FTL, and RL can successfully capture the peak level in the true load pattern (between hour 18 and hour 24). On the second day, IP and FTL anticipated a peak level (between hour 42 and



(a) Weekdays (b) Weekends

Figure 4: Boxplots of MAPEs obtained by CP, IP, FTL, and RL in the four test cases.

Table 1: Summary of the medians of MAPEs. Entries denoted with * indicate that the corresponding values are significantly higher than that of RL in the same column at a confidence level of 95% upon applying the Bonferroni correction for multiple comparisons.

	Weekdays				Weekends			
	8-bus	14-bus	24-bus	123-bus	8-bus	14-bus	24-bus	123-bus
CP	0.325*	0.345*	0.345*	0.337*	0.343*	0.378*	0.334*	0.365*
IP	0.251*	0.301*	0.306*	0.293*	0.298*	0.301*	0.307*	0.326*
FTL	0.268*	0.287*	0.297*	0.280*	0.278*	0.286*	0.294*	0.314*
RL	0.208	0.230	0.258	0.241	0.235	0.216	0.272	0.280

hour 44) close to that observed on the first day (between hour 22 and hour 24). However, the true load pattern changed and a peak level similar to the first day was not reached. We note that IP and FTL missed the true load pattern of the target customer, whereas RL was able to respond more promptly and captured this change. Similar observations regarding the relative performance of IP, FTL, and RL can be made regarding the 8-bus, 14-bus, and 24-bus test cases shown in Fig. 5 as well. A comparison of Fig. 6 for customers with volatile load behaviors with Fig. 5 manifests that the true load patterns shown in Fig. 6 indeed exhibit more rapid changes and higher variability. Nevertheless, earlier observations regarding the relative performance of IP, FTL, and RL still hold; and RL dominates IP and FTL by capturing the true load patterns more promptly and accurately.

We close this section with some reflections on the dominance of the pro-

Table 2: Summary of the medians of MASEs. Entries denoted with * indicate that the corresponding values are significantly higher than that of RL in the same column at a confidence level of 95% upon applying the Bonferroni correction for multiple comparisons.

	Weekdays				Weekends			
	8-bus	14-bus	24-bus	123-bus	8-bus	14-bus	24-bus	123-bus
CP	0.322*	0.343*	0.349*	0.346*	0.424*	0.418*	0.403*	0.436*
IP	0.251*	0.318*	0.319*	0.303*	0.365*	0.345*	0.358*	0.337*
FTL	0.289*	0.307*	0.301*	0.297*	0.342*	0.332*	0.359*	0.345*
RL	0.213	0.231	0.253	0.241	0.291	0.249	0.311	0.299

posed RL-based sampling algorithm. First and foremost, in terms of the objective function, IP aims at maximizing the training sample size that can be obtained for each target customer under the daily bandwidth constraint. FTL intends to minimize the cumulative predictive error in hindsight. In contrast, RL solves an MDP formulated to directly maximize the expected cumulative future predictive accuracy. Second, IP produces a static sampling policy over a prescribed decision horizon, and an update of its sampling policy must be made at the end of a decision horizon. FTL and RL can continually update their sampling policies through real-time data interactions; however, RL outperforms FTL thanks to its capability to optimize with the outcomes of all possible future behaviors taken into account. Therefore, RL is arguably most suitable for implementation in real-life smart grids as it can learn to accommodate volatile load behaviors adaptively and promptly.

6. Conclusions and Future Research

In this paper, we identified opportunities and challenges brought by controlling smart meter sampling rates under some bandwidth constraint for enhancing the overall load forecastability in smart grids. We formulated the sampling-rate control problem as an MDP and developed a novel RL-based algorithm to solve for an optimal sampling-rate control policy. The proposed algorithm can be implemented both offline and online, with the latter capable of performing real-time data interaction with smart grids. Numerical experiments show that the proposed RL-based algorithm can accommodate

volatile load behaviors more promptly and deliver superior predictive performance. As one of the first studies dedicated to the sampling-rate control problem, this work paves the way for future research on utilizing machine learning techniques to achieve more efficient and reliable performance of smart grids.

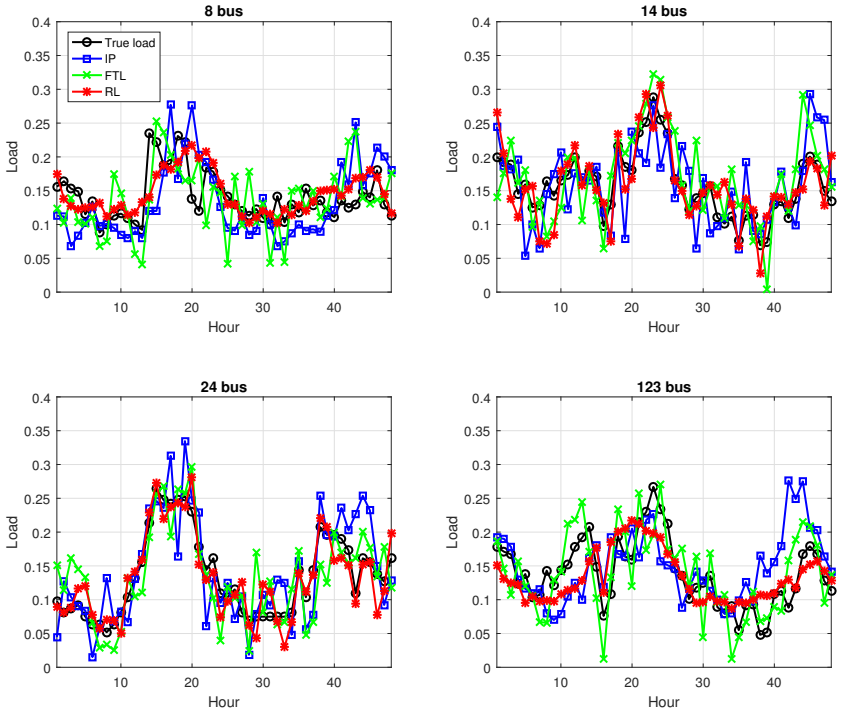


Figure 5: Predictions for a representative customer with stable load behaviors on two consecutive days in each of the four test cases.

We end this work with a discussion on future research directions that would further improve the applicability and efficiency of the proposed sampling-rate control approach in real-world implementations. First, one of the innovations of this work is that the online and offline versions of the RL-based algorithm can meet the needs of different types of customers in the power system. The offline version can be utilized for industrial customers whose

load patterns are stable and infrequent updates of the sampling policy are adequate. The sampling policy obtained from the offline version can remain unchanged for an extended operation period, as long as the predictive performance remains satisfactory. In this case, how to intelligently determine the right timing to run the offline version for policy updates deserves a further investigation. Second, this work focuses on exploiting differences in customer's load behaviors while allocating sampling rates under the bandwidth constraint. For a large-scale distribution grid, it can be more computationally efficient to cluster customers based on similarities in their load behaviors first and then consider sampling-rate control across distinct clusters.

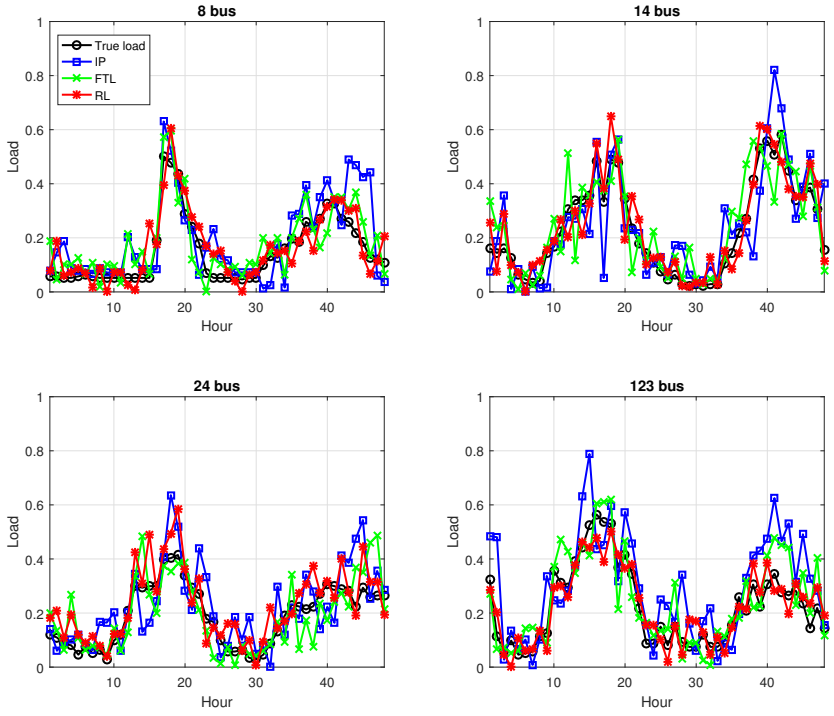


Figure 6: Predictions for a representative customer with volatile load behaviors on two consecutive days in each of the four test cases.

Acknowledgments

The work of the first two authors was supported by the National Science Foundation [grant numbers IIS-1849300, CMMI-1846663].

References

Alamaniotis, M., Chatzidakis, S., & Tsoukalas, L. H. (2014). Monthly load forecasting using kernel based gaussian process regression. In *MedPower* (pp. 1–8).

Alberg, D., & Last, M. (2018). Short-term load forecasting in smart meters with sliding window-based arima algorithms. *Vietnam Journal of Computer Science*, *5*, 241–249.

Amjady, N. (2006). Day-ahead price forecasting of electricity markets by a new fuzzy neural network. *IEEE Transactions on Power Systems*, *21*, 887–896.

Arora, S., & Taylor, J. W. (2018). Rule-based autoregressive moving average models for forecasting load on special days: A case study for france. *European Journal of Operational Research*, *266*, 259–268.

Balachandran, K., Olsen, R. L., & Pedersen, J. M. (2014). Bandwidth analysis of smart meter network infrastructure. In *Proceedings of 16th International Conference on Advanced Communication Technology* (pp. 928–933).

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213.

Bozic, M., Stojanovic, M., Stajic, Z., & Tasic, D. (2013). A new two-stage approach to short term electrical load forecasting. *Energies*, *6*, 2130–2148.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using

RNN Encoder–Decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (pp. 1724–1734).

Claessens, B. J., Vrancx, P., & Ruelens, F. (2018). Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control. *IEEE Transactions on Smart Grid*, *9*, 3259–3269.

Diebold, F. X., & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business and Economic Statistics*, *13*, 253–263.

Ekonomou, L., Christodoulou, C., & Mladenov, V. (2016). A short-term load forecasting method using artificial neural networks and wavelet analysis. *International Journal of Power Systems*, *1*, 64–68.

Elattar, E. E., Goulermas, J., & Wu, Q. H. (2010). Electric load forecasting based on locally weighted support vector regression. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: (Applications and Reviews)*, *40*, 438–447.

Fan, Y., Tian, F., Qin, T., Bian, J., & Liu, T.-Y. (2017). Learning what data to learn. Arxiv.org/abs/1702.08635v1.

Grainger, J. J., & Stevenson, W. D. (1994). *Power System Analysis*. McGraw-Hill Education.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Hoi, S. C., Sahoo, D., Lu, J., & Zhao, P. (2018). Online learning: A comprehensive survey. ArXiv:1802.02871v2[cs.LG].

Jain, R. K., Smith, K. M., Culligan, P. J., & Taylor, J. E. (2014). Forecasting energy consumption of multi-family residential buildings using support vector regression: Investigating the impact of temporal and spatial

monitoring granularity on performance accuracy. *Applied Energy*, *123*, 168–178.

Kell, A., McGough, A. S., & Forshaw, M. (2018). Segmenting residential smart meter data for short-term load forecasting. In *Proceedings of the Ninth International Conference on Future Energy Systems* (pp. 91–96).

Kermanshahi, B. (1998). Recurrent neural network for forecasting next 10 years' loads of nine Japanese utilities. *Neurocomputing*, *23*, 125–133.

Lloyd, J. R. (2014). GEFCom2012 hierarchical load forecasting gradient boosting machines and Gaussian processes. *International Journal of Forecasting*, *30*, 369–374.

Lu, R., Hong, S. H., & Yu, M. (2019). Demand response for home energy management using reinforcement learning and artificial neural network. *IEEE Transactions on Smart Grid*, (pp. 1–11).

New York Independent System Operator (2015). NYISO load data. URL: `http://www.nyiso.com/public/markets_operations/market_data/load_data/index.jsp`.

Nimbargi, S., Mhaisne, S., Nangare, S., & Sinha, M. (2016). Review on AMI technology for smart meter. In *Proceedings of the 2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology* (pp. 21–27).

Nystrup, P., Lindstrom, E., Pinson, P., & Madsen, H. (2020). Temporal hierarchies with autocorrelation for load forecasting. *European Journal of Operational Research*, *280*, 876–888.

PJM (2014). PJM metered load data. URL: `http://www.pjm.com/markets-and-operations/ops-analysis/historical-load-data.aspx`.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, NY.

Rahman, M., & Mto, A. (2013). Investigation of bandwidth requirement of smart meter network using OPNET modeler. *Smart Grid and Renewable Energy*, *4*, 378–390.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA.

Rendon-Sanchez, J. F., & de Menezes, L. M. (2019). Structural combination of seasonal exponential smoothing forecasts applied to load forecasting. *European Journal of Operational Research*, *275*, 916–924.

Renewables.ninja (2017). URL: `https://www.renewables.ninja.`

Ruelens, F., Claessens, B. J., Vandael, S., De Schutter, B., Babuška, R., & Belmans, R. (2017). Residential demand response of thermostatically controlled loads using batch reinforcement learning. *IEEE Transactions on Smart Grid*, *8*, 2149–2159.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. (2nd ed.). The MIT Press, Cambridge, MA.

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems* (pp. 1057–1063).

U.S. Department of Energy (2016). Advanced metering infrastructure and customer systems. URL: `https://www.energy.gov/sites/prod/files/2016/12/f34/AMI%20Summary%20Report_09-26-16.pdf.`

Verizon (2017). Use case: Verizon grid wide smart metering solutions. URL: `https://solutionslab.vzw.com/document/grid-wide-smart-metering-solutions-use-case/.`

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, *8*, 229—-256.

Xie, G., Chen, X., & Weng, Y. (2018). An integrated Gaussian process modeling framework for residential load prediction. *IEEE Transactions on Power Systems*, *33*, 7238–7248.

Zhang, K., Koppel, A., Zhu, H., & Basar, T. (2020). Global convergence of policy gradient methods to (almost) locally optimal policies. *arXiv:1906.08383v3 [math.OC] 28 Jun 2020*, (pp. 1–57).

Zheng, J., Chen, X., Yu, K., Gan, L., Wang, Y., & Wang, K. (2018). Short-term power load forecasting of residential community based on gru neural network. In *2018 International Conference on Power System Technology* (pp. 4862–4868).

Zimmerman, R. D., & Murillo-Sanchez, C. E. (2010). MATPOWER, a MATLAB power system simulation package. URL: `http://www.pserc.cornell.edu/matpower/manual.pdf`.