Adaptive Battery Control with Neural Networks

Fiodar Kazhamiaka*
fkazhami@uwaterloo.ca
University of Waterloo
Waterloo, Canada

Srinivasan Keshav
keshav@uwaterloo.ca
University of Waterloo

Catherine Rosenberg
cath@uwaterloo.ca
University of Waterloo

ABSTRACT

The return on investment of a battery system is maximized if the battery control strategy is appropriately matched to the operating environment (e.g., pricing scheme, electrical load). For residential battery systems, the current practice is to statically determine the control policy prior to system installation; the battery subsequently spends upwards of 10 years operating in a dynamic environment. A state-of-the-art model predictive controller (MPC) can adapt to changes in the system, but is limited by its high online computational requirements. To better extract value at a reasonable online computational cost, we propose an *adaptive* battery controller framework that learns a control strategy by encoding an MPC policy in a neural network, as data becomes available, to adapt the control to the operating environment. We evaluate our controller in the context of a solar PV-storage system deployed in Texas under a time-of-use pricing scheme. We find that our controller gets to within 5-10% of optimal performance, and outperforms a default control strategy for PV-storage systems within a few months of installation.

CCS CONCEPTS

• **Hardware** → **Batteries**; *Renewable energy*; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Deep neural networks, battery control, adaptive system, model predictive control

ACM Reference Format:
Fiodar Kazhamiaka, Srinivasan Keshav, and Catherine Rosenberg. 2019. Adaptive Battery Control with Neural Networks. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems (e-Energy '19), June 25–28, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3307772.3331032

1 INTRODUCTION

In the last decade, steadily declining prices of Lithium-ion batteries and solar photovoltaic (PV) cells [10, 34] have made it economically advantageous for residential and commercial buildings in some jurisdictions to generate and store their own energy, and thereby reduce their dependency on the centralized grid [9, 20]. Meeting a significant fraction of electricity demand in a building via renewable sources requires large battery systems, which represent a substantial monetary investment. Hence, many studies have focused on increasing the return on investment of these systems, by sizing and controlling them appropriately based on their operating environment [6, 19, 29], and using them for several applications concurrently (this is called application stacking)[15, 33].

The current practice for residential battery system control is to deploy them with a static control policy, with no further modifications over the life of the system. However, batteries have a long lifetime, with many warranties lasting 5-10 years. Hence, the operating environment may change over the system lifetime, and this may require adjustments to the control strategy to make it effective. For example, changes to the grid pricing scheme[1], electrical load (new appliances), and stacking of additional applications are possible. Considering this dynamic operating environment, static rules may be unable to efficiently control the battery over its lifetime. Designing new rules to adapt to the changing environment is a prohibitively tedious task given the variety in the combinations of load patterns, grid pricing schemes, and (stacked) applications that require tailored control rules.

The state of the art in control of battery systems is model predictive control (MPC) [4, 18, 24, 40]. An MPC can be adapted relatively painlessly to a changing operating environment by updating its predictive models to match the changes in the patterns of load and generated power. Changes to the grid pricing and new applications can be also be accommodated by updating the underlying optimization objective function and system model. However, an MPC's high computational cost can make it impractical to use for residential battery systems. Our key insight is that, to circumvent this problem, deep neural networks can be trained to approximate the control policy of an MPC at a fraction of the online computational cost [16, 46].

The limiting factor of using a deep neural network for residential battery system control is that training the network requires data about the operating environment, which is typically unavailable at the time of system deployment. However, this data *can* be collected by the system post-deployment. To account for this limitation, we have designed a controller that makes periodic updates its underlying neural network. We present a preliminary evaluation of our approach for a residential system comprised of PV panels and a battery, in an environment where time-of-use grid pricing is used, load patterns can change over time, and there is an option to sell PV power to the grid.

Our contributions are as follows:

- We have developed an adaptive battery controller framework that uses a combination of model predictive control

e-Energy '19, June 25–28, 2019, Phoenix, AZ, USA

ACM ISBN 978-1-4503-6671-7/19/06…$15.00
https://doi.org/10.1145/3307772.3331032

[1] The prices and price periods in Ontario's time-of-use electricity pricing scheme have changed at least once a year in the recent past.

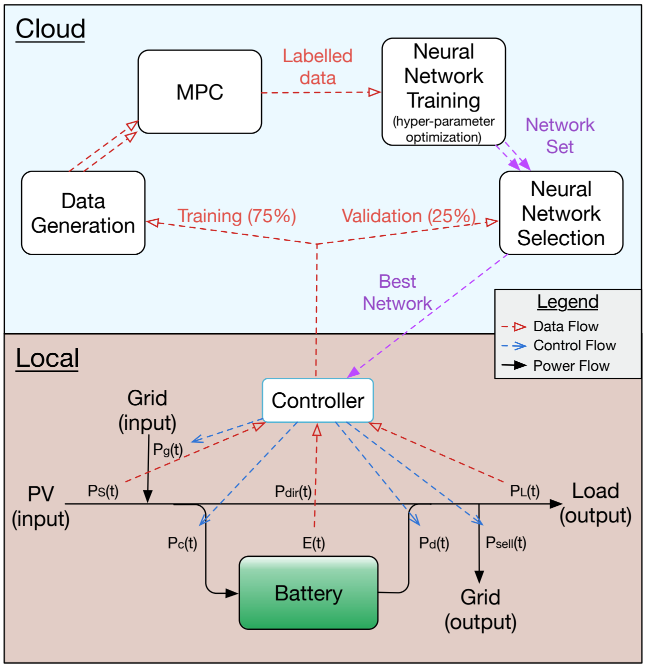

Figure 1: System diagram

(MPC), neural networks, and simulations to optimize battery operation over time.
- We show that our controller can outperform the default static operating strategy used in PV-storage systems within a few months of deployment and approaches MPC-level performance at a fraction of the online computational cost.

The rest of the paper is organized as follows. In Section 2, we describe the design of the controller. In Section 3, we evaluate the performance of our controller and compare it to the default control in most battery systems, an MPC-based control and an Oracle. In Section 4, we discuss the current approaches to battery system control and how they relate to our vision of an adaptive battery system. We discuss our future research plans and conclude the paper in Section 5

2 DESIGN

In this section, we describe the context for our system and the controller architecture. The system configuration is shown in the lower half of Figure 1, which is installed in a home and is composed of the following:

(1) an array of solar PV panels
(2) a Lithium-ion battery
(3) a bi-directional connection to the local electricity grid to buy and sell electricity
(4) a connection to the residential electricity load
(5) a controller

Our design goals are as follows:

- **High performance**. The controller should outperform the default control scheme (lower benchmark), and approach the performance of MPC (upper benchmark) at a fraction of the online computational cost.
- **Minimizing offline computation cost**; due to the costs associated with cloud computing, we want to minimize the amount of computation required to achieve a high level of performance.
- **Flexibility** to adapt to different system configurations, operating environments, and applications with minimal user effort and interaction.

2.1 Local system

The purpose of the system is to minimize the cost of providing power to the home, which can also be offset by selling energy. The main control variables are the following:

- power used to charge the battery (P_c)
- power discharged from the battery (P_d)

The power purchased from the grid (P_g) and sold to the grid (P_{sell}) can thus be directly computed to balance the power in the system. The prices for buying electricity are set according to a time-of-use pricing scheme. We assume the system has access to the grid pricing scheme, which is published by the electrical utility company.

Initially, the system is operated using a default control strategy for PV-storage systems which is described in Algorithm 1.

Data: The current solar ($P_S(t)$) and load ($P_L(t)$) measurements
if $P_S(t) > P_L(t)$ **then**
    Charge the battery as much as possible;
    Sell the excess solar;
else
    Discharge the battery to make up for the difference between $P_S(t)$ and $P_L(t)$;
    Purchase power from the grid if the battery power was not enough;
end

Algorithm 1: Default control strategy.

Immediately after the battery system is installed, it begins to collect data on solar PV generation (P_S) and household electricity load (P_L). After a set amount of time has passed, or when prompted by the system owner, the collected data is sent to a server in the cloud and used to train a neural network. If training is successful, i.e., the network is validated to outperform the default strategy on historical observations via system simulation, then the network is uploaded into the system controller. The controller then uses the network to decide how to charge and discharge the battery, applying a post-processing step to the network output to ensure that the constraints of the system are met (see Section 2.6 for details). The network is retrained once a significant amount of additional data is collected, and the old network is replaced by the new one if it is determined to have superior performance via simulations.

2.2 Neural network architecture

The most common type of learning studied for battery system applications is deep reinforcement learning [11, 12, 28]. However, these networks are notoriously difficult to tune, require large amounts of data and computational resources to converge on an effective

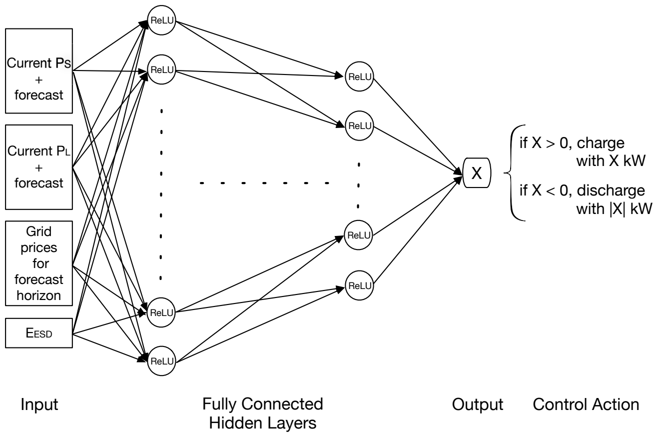

Figure 2: Network architecture

solution, and our initial attempts with RL algorithms for this problem were unsuccessful. Hence, in our controller design, we forego reinforcement learning in favour of a supervised learning approach via deep feed-forward networks [13], which are trained in the cloud using an MPC to label the collected data with good control actions. We discuss why an MPC is not ideal for directly controlling the network in Section 4.

The inputs to the controller can be any data related to the system. In the case of our test system, the following inputs are available:

- Current PV generation, $P_S(t)$
- Current load, $P_L(t)$
- Current battery energy content, $E_{ESD}(t)$
- Grid prices[2], $[\pi_g(t) \dots \pi_g(t + T_h)]$
- PV generation predictions, $[P_S(t + 1) \dots P_S(t + T_h)]$
- Load predictions, $[P_L(t + 1) \dots P_L(t + T_h)]$

Battery control is a regression problem, where the output of the network is the charging or discharging rate (recall that once this is known, we can compute the amount of power to sell or to buy), which we model as a single neuron where a negative output indicates discharging and a positive output indicates charging actions. The magnitude of the output neuron's value is the charging/discharging rate. Figure 2 illustrates the network's structure.

The main hyper-parameters of the network are the number of hidden layers, number of nodes per hidden layer, and number of training epochs. Our training algorithm explores different combinations of hyper-parameters to find the most effective configuration. We use the rectified linear unit (ReLU) activation function for each neuron [3], and gradually decrease the number of neurons of deeper hidden layers in order to lower the training time. The output neuron does not have an activation function, which allows for the output to be a real number.

The standard practice for dealing with neural network inputs in the form of time series, such as the future grid prices and PV/load predictions, is to use convolutional filter layers to extract features from each series [35]. The cost of using 1-D convolutional layers would have to be balanced with the cost of searching through the space of other network hyper-parameters. The results presented in this paper use networks which do not have any convolutional layers, since we found that they were not necessary to achieve effective control performance for the system under study, though they may become necessary for other systems.

[2]For residential consumers, grid prices are typically available months in advance. We found that giving 24 hours of look-ahead was enough to get a satisfactory control performance for the system under study.

2.3 Training data

Deep neural networks perform better when a large amount of training data is available. The amount of time needed to collect a sufficiently large amount of data after system installation can be prohibitive to the adaptive neural network approach. To get around this issue, it is possible to synthesize additional data by using model-based approaches such as ARMA or Gaussian Mixture Models. In this paper, we use the following simple method to expand the amount of PV generation and load data available for training:

(1) Start with original training set of size N
(2) Generate a vector of N *perturbation factors* by conducting a bounded random walk in the range 0.95 and 1.05 with a step size of 0.01
(3) Take an element-wise product of the training set and the perturbation factors vector, and add the resulting dataset to the expanded training set
(4) Repeatedly perturb the original training set until sufficient amount of training data has been generated.

The perturbed data is heavily correlated to the original dataset, and doesn't add completely unseen data patterns to the training set. Nevertheless, our approach creates altered sets of PV and load data that resemble the original measurements (within 5%) and preserves the continuous structure of data over time, while being sufficiently different to provide the network with "fresh" data to train on. In our testing, we have observed that increasing the size of the training dataset by 3-5× with this data generation approach improves the stability of the control actions and the control performance of the network in the first year of system deployment where the original training dataset is small.

Note that we do not use all of the collected data for training; 25% of the data is reserved for network selection, in which we validate the performance of the selected network. This data is labelled with good control actions using an MPC, as discussed in the next subsection.

2.4 MPC

One of the most crucial aspects of our approach is labelling the training data with *effective* control actions. In general, a neural network could be used to approximate any practical control strategy, in the sense that the strategy uses only practically-available information to compute the control actions. The inputs of the strategy map to the inputs of the neural network. We note that using the true optimal control for labelling is impractical because it may depend on perfect knowledge of the entire operating horizon, which is typically impossible to obtain in an online control setting.

To label the data with control actions, we formulate an MPC, i.e., a discrete mathematical optimization problem for optimal system control over a given time horizon. We refer to Figure 1 which shows the label on each power flow, as a guide for some of our notation.

The length of the MPC horizon is T_h time slots, and each time slot is of length T_u. The power flowing through the system in each time slot t is constant. Due to conservation of power, the amount of PV generation and purchased grid power cannot exceed the sum of power used to charge the battery and the power that flows directly to meet the load or be sold ($P_{dir}(t)$). Likewise, the amount of power being sold and demanded by the load cannot exceed the sum of $P_{dir}(t)$ and the amount of power being discharged.

The battery has physical constraints which are described in our model. We use the Lithium-ion battery model developed and validated in [22] for optimization problems, and we briefly describe it here. The battery energy content at the end of time slot t is denoted $E_{ESD}(t)$. The limits on battery energy content have a linear relationship with the charging/discharging power being applied, as described by the u_1, u_2, v_1, and v_2 parameters. The battery also has limits on the charging and discharging power, denoted α_c and α_d respectively. A fraction of power is lost when the battery is charged and discharged due to imperfect efficiency of power conversion, and the remaining fraction is denoted η_c for charging and η_d for discharging. The battery cannot be charged and discharged simultaneously, which also needs to be enforced in our model.

We denote $\pi_g(t)$ and $\pi_{sell}(t)$ to be the price for buying and selling energy in time slot t, respectively. They are input to the problem.

Combining our system constraints, we formulate the problem of minimizing the electricity payment of the system owner as an integer linear program at time t:

Given $[P_S(t), \ldots, P_S(t+T_h)]$, $[P_L(t), \ldots, P_L(t+T_h)]$, the battery parameters: $\alpha_c, \alpha_d, \eta_c, \eta_d, u_1, u_2, v_1, v_2$, the battery content at time t: U_t, the prices $[\pi_g(t), \ldots, \pi_g(t+T_h)]$, $[\pi_{sell}(t), \ldots \pi_{sell}(t+T_h)$), as well as T_u and T_h:

$$\min_{\substack{P_{dir}(k), P_c(k), P_d(k),\\ P_g(k), P_{sell}(k), I(k)}} \sum_{k=t}^{t+T_h} (\pi_g(k)P_g(k) - \pi_{sell}(k)P_{sell}(k))T_u \tag{1}$$

subject to

$$0 \leq P_g(k), P_{dir}(k), P_{sell}(k) \qquad \forall k \in [t, t+T_h] \tag{2}$$

$$P_{dir}(k) + P_d(k) = P_L(k) + P_{sell}(k) \qquad \forall k \in [t, t+T_h] \tag{3}$$

$$E_{ESD}(t) = U_t \tag{4}$$

$$E_{ESD}(k+1) = E_{ESD}(k) + \eta_c P_c(k) T_u - \frac{P_d(k)}{\eta_d} T_u \qquad \forall k \in [t, t+T_h] \tag{5}$$

$$0 \leq P_c(k) + P_{dir}(k) \leq P_S(k) + P_g(k) \qquad \forall k \in [t, t+T_h] \tag{6}$$

$$0 \leq P_c(k) \leq I(k)\alpha_c \qquad \forall k \in [t, t+T_h] \tag{7}$$

$$0 \leq P_d(k) \leq (1 - I(k))\alpha_d \qquad \forall k \in [t, t+T_h] \tag{8}$$

$$I(k) \in \{0, 1\} \qquad \forall k \in [t, t+T_h] \tag{9}$$

$$u_1 P_d(k) + v_1 \leq E_{ESD}(k+1) \leq u_2 P_c(k) + v_2 \qquad \forall k \in [t, t+T_h] \tag{10}$$

where $I(k)$ is a binary integer which prevents simultaneous charging and discharging. It can be shown that the optimal solution will not have simultaneous charging and discharging due to the inefficiencies in the battery, which means the binary integer $I(k)$ can be removed and the problem is simplified to a linear program (LP).

This LP is used as part of an MPC, which is based on iterative optimization of the system control over a finite time horizon. At time t, the control action is decided by solving the optimization problem in order to determine the control actions that optimize the objective function over a finite time horizon $[t, T_h]$. The first control action (corresponding to time t) is implemented, and the process is repeated in the next control time slot $t+1$ with an updated $U_t = E(t)$. In time slot t, $P_L(t)$ and $P_S(t)$ represent the measured load and PV generation, respectively, while $[P_L(t+1) \ldots P_L(t+T_h)]$ and $[P_S(t+1) \ldots P_S(t+T_h)]$ are predictions.

The inputs to the neural network are exactly the same as the inputs to the MPC. We solve the MPC repeatedly to label every time slot of solar, load, and initial battery state data with the charging and discharging actions computed by the MPC. The inputs at time slot t are assigned the label $P_c(t) - P_d(t)$ as computed by the MPC. In each iteration of MPC, the battery energy content is updated to reflect the action taken in the previous iteration. The labelled dataset is used to train neural networks.

A neural network controller trained using MPC can adapt to system changes that are expressed in the PV generation and load data without any modifications to the controller. Changes to the system, grid pricing scheme, and applications are communicated to the controller via modification of the underlying optimization problem of the MPC. The MPC labels the data with updated control actions by which the neural network learns how to operate in the new environment. We note that a linear program can be solved even with basic computational hardware, and a linear MPC could be used to operate the system directly without a neural-network approximation. We use the linear problem as a first step to understand the potential of this approach, since it greatly simplifies and speeds up the computation of our preliminary analysis, and discuss our future plans in section 5.

2.5 Training

In this step, we train many different neural networks, each with a different structure, i.e., different numbers of layers and nodes per layer. Each one of the network configurations is trained using the back-propagation training algorithm with a mean-squared error cost function for minimizing the difference between the network output and the MPC actions computed for the given input. During training, we use a dropout rate of 0.2 at each hidden layer to help prevent overfitting [38].

The network is trained with a set number of epochs (i.e., passes through the training data), and the network weights at the end of each epoch are saved. After training is completed, the set of networks are tested to select the network which will be uploaded to the controller.

2.6 Selection

To validate the performance of the trained networks, we run a simulation for each network in which the network is used to make control decisions on the system. The system model is exactly the same as the one used in LP formulation (Section 2.4).

The selection is done in two steps.

Step 1: All of the newly trained networks are simulated on 25% of the data that was with-held from training, to single out the

Table 1: Battery model parameters

Parameter	α_c	α_d	u_1	u_2	v_1	v_2	η_c	η_d^*
House 1	8	8	0.053	-0.125	0	8	0.99	1.11
House 2	8	8	0.053	-0.125	0	8	0.99	1.11
House 3	14	14	0.053	-0.125	0	14	0.99	1.11

*includes inverter inefficiencies of ~10%

network with the most effective performance on unseen data.

Step 2: The best network from Step 1 is compared against the network that is currently deployed, by simulating them both on all of the data collected so far; if there is no network currently deployed, the new network is compared against the default control strategy. If the new network is the winner, it is uploaded to the controller.

Note that in both the simulations and at the controller, the output of the network is processed by an algorithm to ensure that the physical constraints of the system are met before the control action is taken. First, the algorithm ensures that the constraints of the battery are being met, as described by Constraints 7, 8, and 10 in the LP, by decreasing the charging/discharging action if necessary. Next, the power in the system is balanced (Constraints 2 and 3) by selling less and then buying more power from the grid if there is a power deficit, or buying less and then selling more if there is a power surplus.

3 EVALUATION

We use four years of PV generation and load data obtained from three houses in the Pecan Street Dataport [1] to simulate different realizations of this system. The hourly data was collected from Texas, USA, over a period of four years. All three homes have high electricity consumption in the range of 35-45 kWh per day, making them good candidates for a PV panel and battery installation. The night grid price is set to \$0.04 per kWh, and the higher day price to \$0.16 per kWh between October and May, and \$0.21 per kWh between June and September, reflecting a ToU pricing scheme offered by a Texas utility company [43]. The price for selling electricity is set to a constant value of \$0.03 per kWh, which is just below the off-peak grid price.

We compare our adaptive controller, the default control algorithm, and the MPC in terms of the grid cost paid by the home owner. We also compare these costs to the lowest possible (optimal) cost obtained via an Oracle, which we compute by solving the LP in Section 2.4 over the entire four year horizon (T_h = 4 years). We use perfect hourly forecasts ($T_u = 1$) with $T_h = 24$ as input for the MPC and neural network simulations, deferring an evaluation with realistic prediction errors and optimization of the prediction horizon to future work. The parameters used in our system model are given in Table 1.

The network training process was activated on the following days: 10, 20, 40, 80, 120, 190, 365, 550, 730, 910, 1090, 1270, and 1450. During the data generation phase, the amount of training data was increased by a factor of 4. During the training phase, we looked at networks with between 3 and 7 hidden layers, 40 and 100 nodes per layer, and 21 epochs; a total of 24 different network structures are tested at each retraining period, each with 21 different sets of weights (one for each training epoch), giving a total of 504 different networks to choose from in the selection step.

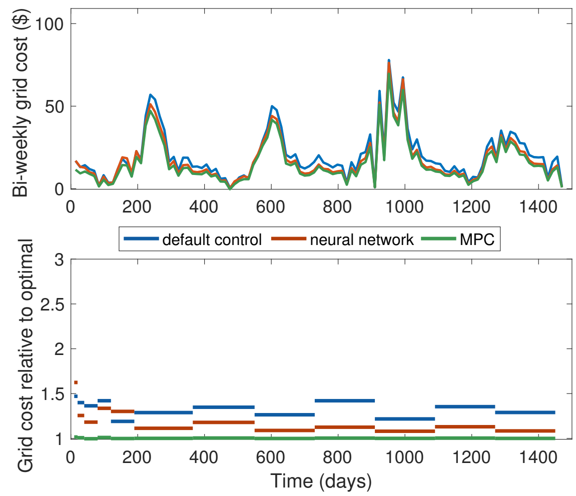

Figure 3: House 1: bi-weekly grid cost, and the total cost between re-training periods relative to the optimal cost.

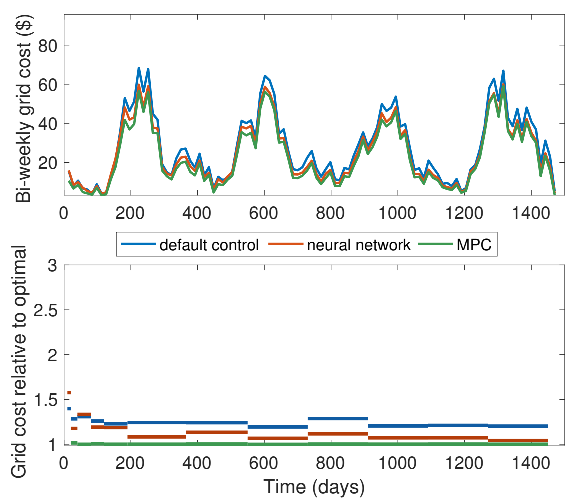

Figure 4: House 2: bi-weekly grid cost, and the total cost between re-training periods relative to the optimal cost.

The results for the three houses are summarized in Figures 3, 4, and 5, respectively. Each figure shows the bi-weekly grid cost over four years for the three control methods, as well as a comparison of their relative performance with respect to the Oracle. Across all three houses, we see that MPC is quasi-optimal. The neural network is able to get within 25% of the optimal cost 200 days after deployment, and within 5-10% in 2-4 years. For houses 1 and 2, the neural network outperforms the default strategy within 100 days of system deployment. Figure 5 also shows the changes in

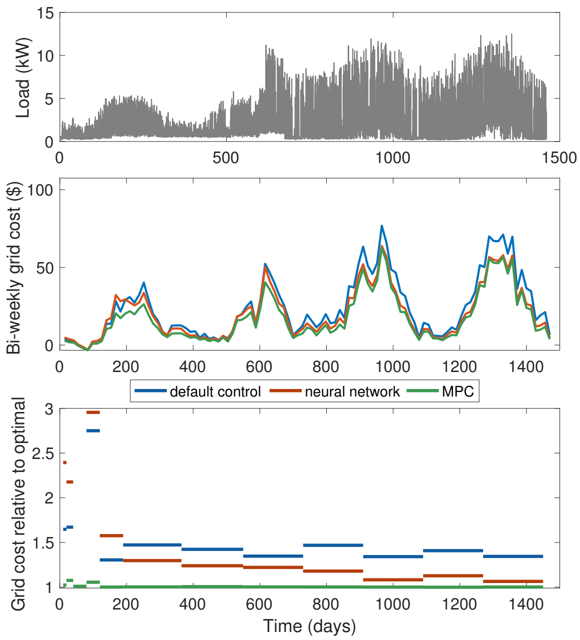

Figure 5: House 3: load, which has large changes halfway through year 2, the biweekly grid cost, and the total cost between re-training periods relative to the optimal cost.

the load pattern approximately halfway through the second year of deployment. Across all houses, the adaptive controller saved between 13-19% on grid costs over the 4 years compared to the default strategy, with savings as high as 25% for the third house after two years of data was collected. In additional tests, we observed that the networks trained on data prior to the load change performed poorly after the change, which highlights the value of re-training the controller on recently-collected data.

We observe a general trend in the size of the most effective networks across different training periods. Smaller networks, i.e, those with fewer nodes and hidden layers, have better performance when there is not much data available, while larger networks eventually perform better as more training data becomes available. To reduce the amount of computation done during the training of the network set, one possible optimization is to check for convergence of the structure of the best network, and remove structures with historically poor performance from the set.

4 RELATED WORK

In this section we discuss three methods for battery control: MPC, rule-based control, and neural networks.

4.1 Model Predictive Control (MPC)

MPC is commonly used for industrial process control [45], although there have been applications of MPC for a variety of control problems, such as optimizing energy efficiency [37] and HVAC systems [17] in buildings, and controlling battery and renewable energy systems [44].

MPC has been studied as a control method for energy systems with storage. Teleke et al. [40] consider a system consisting of a wind farm and energy storage that is used to make wind power more dispatchable. One of the observations in this work is the system performance with respect to the length of the time horizon considered for optimization; horizon values ranging from 100 seconds to 30 minutes were tested, showing significant performance benefits from having longer – and hence, more computationally expensive – horizons.

Khalid et al. [23, 24] also consider a wind farm with energy storage but with different objectives. In [23], energy storage is controlled using an MPC approach to reduce the ramping of system power output, and in [24] the system is operated to maximize revenue from selling energy in an electricity market with dynamic pricing. In both cases, it is shown that the MPC can effectively utilize the battery to meet the objectives of the applications. The optimization horizon in [24] is limited to 15 minutes to limit the computational burden.

MPC could be used as the base for an adaptive battery controller, since it requires only a dynamic model of the system and a predictive model for the system environment. The predictive models could be trained as more data is collected by the system to improve the MPC performance, and new applications or pricing schemes could be handled by updates to the optimization objective function and constraints. However, MPC is limited in its applicability by the cost of rapidly solving optimization problems to compute the control action at any given time. Although significant improvements to computation time can be made by improvements to the solving algorithm [26, 44], the hardware requirements of the solver for a non-linear control problem may still dominate any improvements in control effectiveness for small-scale residential system applications; for such applications, a control solution with minimal online computational costs is preferred.

In our work, we train neural networks to approximate the results of the MPC to bypass the heavy online computation. For comparison, the neural networks trained in our evaluation using Python's Keras deep learning library on a single 1.6 GHz CPU were on average 22× faster than a linear MPC running using the CPLEX LP solver [8] on a machine with up to 24 3.0 GHz CPU cores for parallelization and 500 GB of RAM. Using a non-linear MPC – in the case where a more accurate non-linear battery model is needed or if the grid pricing scheme or application constraints could not be expressed linearly – would require at least a few orders of magnitude more time, making it too slow for online control using the limited hardware that is typically available in a battery controller.

4.2 Rule-based control

The rule-based control approach involves creating a set of control rules which relate system state to control actions. Rule-based control requires very low online computational effort, but it is often difficult to come up with rules that result in good system performance. Rules can be specified in explicit "white box" form such as an algorithm or knowledge-based system [39]. The creation of explicit rules is often done by domain experts.

It is common for rule-based strategies to focus more on meeting constraints rather than optimization objectives. In [42], a very simple algorithm is used to control an off-grid system composed of a wind turbine, PV panels, hydrogen fuel cell, and household loads. The algorithm chooses between two control actions depending on whether or not there is excess renewable generation. A similar system was considered in [2], where the control of the system included a heuristic that kept the charge of the battery at high levels to reduce the problems in power balance caused by intermittent renewable sources.

While rule-based approaches have shown to be effective, requiring a domain expert to tweak the battery control rules to adapt to changes is not practical. Furthermore, there is a large number of grid pricing schemes; each utility company offers several options to its residential customers, and an even larger variety is available to commercial and industrial customers. In a general dynamic pricing scenario, it has been shown that optimal policy can be based on energy thresholds [14], i.e., the charge/discharge policy can be described through a relationship between the energy content in the battery, some optimal energy thresholds, and the current state of the system. In [21], we designed efficient operating threshold-based rules for two system deployment scenarios that involve a combination of energy arbitrage, curtailment avoidance, and power leveling applications. Our approach uses heuristics that were obtained from studying the results of offline optimal operation. In this paper, we automate the learning of effective control rules by using artificial neural networks in place of domain experts.

4.3 Neural Networks

Artificial neural networks have been used to control energy systems such as hybrid-electric vehicles and wind farms with energy storage [5, 27, 32]. In [32], a neural network approach to controlling a supercapacitor in a hybrid-electric vehicle showed substantial improvement in energy efficiency compared to other approaches. In [5], a neural network is shown to decrease the cost of operation compared to simple rule-based control and fuzzy logic control. In both studies, the neural network was trained using datasets of "good" control actions which were obtained by using offline optimization methods.

Deep learning methods have shown promise in complex control problems. Deep neural networks have been trained to successfully perform physical tasks such as walking and car driving in a simulated environment [25], play video games [30, 31], and replace PID control for a DC motor [7]. These problems have high-dimensional features (input) that are mapped to a comparatively small number of control variables (output); energy storage control has the same challenge, where a large number of environment variables can be used to decide the battery's charging or discharging rate. Applying deep neural networks to energy systems has been studied in [11] and [28], and recently the concept of supporting and even replacing human effort via artificial neural networks in the management of power systems has been proposed [41].

The online computational cost of using a deep neural network is very low, making it more suitable for small-scale residential systems compared to MPC. The high computational cost of using MPC was handled in a similar way in [46] and [16], where neural networks were used to encode an online MPC policy to control drones and other autonomous robots. The shortcomings are the cost of training the network, and the need for large amounts of data to learn from.

Neural networks offer a natural approach to adaptive system control; automatically re-training the control network with newly-available data allows us to bypass most of the work involved in creating new control rules in response to changes in the system. We believe that a deep learning control framework has the potential to be applicable for a wide combination of energy system deployment and environment factors, including but not limited to variability in the following system characteristics:

- Battery size
- Battery chemistry, including Lithium-ion and other battery technologies
- Application, including stacked applications
- Operating climate, in cases where batteries are combined with renewable sources that depend on wind speed and solar radiation
- Grid pricing for the buying and selling of energy, in cases where the application interacts with the energy grid

5 FUTURE WORK AND CONCLUSION

In this paper, we have developed an adaptive controller framework for battery systems based on neural networks, MPC, and system simulation. Unlike the static algorithms typically used in battery systems, our controller can adapt to changes in the system. We have evaluated our controller on a PV-battery system with time-of-use pricing, and shown that it approaches MPC-level performance at a fraction of the online computational cost of MPC.

Our approach has two limitations. First, there is a cost to train the neural network in the cloud, and this cost has to be balanced with the gains in control performance achieved by training on new data. Second, some time must pass to collect enough data to train an effective network, and during this time the control policy may be ineffective.

In future work, we plan to explore the following:

- The effect of prediction errors. Our current results were computed with perfect predictions for day-ahead PV and load. MPC relies on accurate predictions of the operating environment to make effective control decisions; it is unclear to what degree prediction errors will affect the performance of the neural network. Our plan is to conduct a sensitivity analysis on the impact of prediction errors.
- Transfer learning [36], by training networks on available data collected by other systems to increase the rate at which the network performance improves.
- Test on non-linear systems. Showing that it works on a linear system was the first step of our evaluation. For such systems, it may be possible to use MPC directly rather than using a neural network to encode a the MPC policy, because solving small LPs online can be done efficiently. For systems, applications, and pricing schemes with poor linear approximations, the computational cost of solving non-linear optimization problems online is prohibitive, and encoding the control strategy (in a neural network) to lower the online computational cost is an attractive solution.

- Test on different pricing schemes. We are currently testing on a peak-pricing scheme, where the price increases substantially if grid power usage exceeds a threshold.

REFERENCES

[1] 2018. Pecan Street Inc. Dataport.
[2] Kodjo Agbossou, Mohanlal Kolhe, Jean Hamelin, and Tapan K Bose. 2004. Performance of a stand-alone renewable energy system based on energy storage as hydrogen. *IEEE Transactions on Energy Conversion* 19, 3 (2004), 633–640.
[3] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. 2016. Understanding deep neural networks with rectified linear units. *arXiv preprint arXiv:1611.01491* (2016).
[4] Kyri Baker, Junyao Guo, Gabriela Hug, and Xin Li. 2016. Distributed MPC for efficient coordination of storage and renewable energy sources across control areas. *IEEE Transactions on Smart Grid* 7, 2 (2016), 992–1001.
[5] Ted KA Brekken, Alex Yokochi, Annette Von Jouanne, Zuan Z Yen, Hannes Max Hapke, and Douglas A Halamay. 2011. Optimal energy storage sizing and control for wind power applications. *IEEE Transactions on Sustainable Energy* 2, 1 (2011), 69–77.
[6] Claudio Brivio, Stefano Mandelli, and Marco Merlo. 2016. Battery energy storage system for primary control reserve and energy arbitrage. *Sustainable Energy, Grids and Networks* 6 (2016), 152–165.
[7] Kangbeom Cheon, Jaehoon Kim, Moussa Hamadache, and Dongik Lee. 2015. On Replacing PID Controller with Deep Learning Controller for DC Motor System. *Journal of Automation and Control Engineering Vol* 3, 6 (2015).
[8] IBM ILOG CPLEX. 2009. V12. 1: UserâĂŹs Manual for CPLEX. *International Business Machines Corporation* 46, 53 (2009), 157.
[9] Federica Cucchiella, Idiano DâĂŹAdamo, Massimo Gastaldi, and Vincenzo Stornelli. 2018. Solar photovoltaic panels combined with energy storage in a residential building: An economic analysis. *Sustainability* 10, 9 (2018), 3117.
[10] David Feldman. 2014. Photovoltaic (PV) pricing trends: historical, recent, and near-term projections. (2014).
[11] Vincent François-Lavet, David Taralla, Damien Ernst, and Raphaël Fonteneau. 2016. Deep reinforcement learning solutions for energy microgrids management. In *European Workshop on Reinforcement Learning (EWRL 2016)*.
[12] Mevludin Glavic, Raphaël Fonteneau, and Damien Ernst. 2017. Reinforcement learning for electric power system decision and control: Past considerations and perspectives. *IFAC-PapersOnLine* 50, 1 (2017), 6918–6927.
[13] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.
[14] Pavithra Harsha and Munther Dahleh. 2015. Optimal management and sizing of energy storage under dynamic pricing for the efficient integration of renewable energy. *IEEE Transactions on Power Systems* 30, 3 (2015), 1164–1181.
[15] Holger Hesse, Michael Schimpe, Daniel Kucevic, and Andreas Jossen. 2017. Lithium-ion battery storage for the gridâĂŤa review of stationary battery storage system design tailored for applications in modern power grids. *Energies* 10, 12 (2017), 2107.
[16] Noriaki Hirose, Ryosuke Tajima, and Kazutoshi Sukigara. 2018. MPC policy learning using DNN for human following control without collision. *Advanced Robotics* 32, 3 (2018), 148–159.
[17] Milan Jain, Rachel K Kalaimani, Srinivasan Keshav, and Catherine Rosenberg. 2018. Using personal environmental comfort systems to mitigate the impact of occupancy prediction errors on HVAC performance. *Energy Informatics* 1, 1 (2018), 60.
[18] Jiangliang Jin and Yunjian Xu. 2017. Optimal storage operation under demand charge. *IEEE Transactions on Power Systems* 32, 1 (2017), 795–808.
[19] Fiodar Kazhamiaka, Yashar Ghiassi-Farrokhfal, Srinivasan Keshav, and Catherine Rosenberg. 2018. Robust and Practical Approaches for Solar PV and Storage Sizing.. In *e-Energy*. 146–156.
[20] Fiodar Kazhamiaka, Patrick Jochem, Srinivasan Keshav, and Catherine Rosenberg. 2017. On the influence of jurisdiction on the profitability of residential photovoltaic-storage systems: A multi-national case study. *Energy Policy* 109 (2017), 428–440.
[21] Fiodar Kazhamiaka, Catherine Rosenberg, and Srinivasan Keshav. 2016. Practical strategies for storage operation in energy systems: design and evaluation. *IEEE Transactions on Sustainable Energy* 7, 4 (2016), 1602–1610.
[22] Fiodar Kazhamiaka, Catherine Rosenberg, Srinivasan Keshav, and Karl-Heinz Pettinger. 2016. Li-ion storage models for energy system optimization: the accuracy-tractability tradeoff. In *Proceedings of the Seventh International Conference on Future Energy Systems*. ACM, 17.
[23] M Khalid and AV Savkin. 2010. A model predictive control approach to the problem of wind power smoothing with controlled battery storage. *Renewable Energy* 35, 7 (2010), 1520–1526.
[24] Arash Khatamianfar, Muhammad Khalid, Andrey V Savkin, and Vassilios G Agelidis. 2013. Improving wind farm dispatch in the Australian electricity market with battery energy storage using model predictive control. *IEEE Transactions on Sustainable Energy* 4, 3 (2013), 745–755.
[25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
[26] Stefano Longo, Eric C Kerrigan, Keck Voon Ling, and George A Constantinides. 2011. A parallel formulation for predictive control with nonuniform hold constraints. *Annual Reviews in Control* 35, 2 (2011), 207–214.
[27] Srdjan M Lukic, Jian Cao, Ramesh C Bansal, Fernando Rodriguez, and Ali Emadi. 2008. Energy storage systems for automotive applications. *IEEE Transactions on industrial electronics* 55, 6 (2008), 2258–2267.
[28] Brida Mbuwir, Frederik Ruelens, Fred Spiessens, and Geert Deconinck. 2017. Battery energy management in a microgrid using batch reinforcement learning. *Energies* 10, 11 (2017), 1846.
[29] Adel Merabet, Khandker Tawfique Ahmed, Hussein Ibrahim, Rachid Beguenane, and Amer MYM Ghias. 2017. Energy management and control system for laboratory scale microgrid based wind-PV-battery. *IEEE Transactions on Sustainable Energy* 8, 1 (2017), 145–154.
[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.
[32] Jorge Moreno, Micah E Ortúzar, and Juan W Dixon. 2006. Energy-management system for a hybrid electric vehicle, using ultracapacitors and neural networks. *IEEE transactions on Industrial Electronics* 53, 2 (2006), 614–623.
[33] Jeremy Neubauer and Ahmad Pesaran. 2011. The ability of battery second use strategies to impact plug-in electric vehicle prices and serve utility energy storage applications. *Journal of Power Sources* 196, 23 (2011), 10351–10358.
[34] Björn Nykvist and Måns Nilsson. 2015. Rapidly falling costs of battery packs for electric vehicles. *Nature Climate Change* 5, 4 (2015), 329–332.
[35] Keiron O'Shea and Ryan Nash. 2015. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).
[36] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.
[37] Tim Salsbury, Prashant Mhaskar, and S Joe Qin. 2013. Predictive control methods to improve energy efficiency and reduce demand in buildings. *Computers & Chemical Engineering* 51 (2013), 77–85.
[38] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[39] Rudi Studer, V Richard Benjamins, and Dieter Fensel. 1998. Knowledge engineering: principles and methods. *Data & knowledge engineering* 25, 1 (1998), 161–197.
[40] Sercan Teleke, Mesut E Baran, Subhashish Bhattacharya, and Alex Q Huang. 2010. Optimal control of battery energy storage for wind farm dispatching. *IEEE Transactions on Energy Conversion* 25, 3 (2010), 787–794.
[41] Nikita Tomin, Victor Kurbatsky, and Michael Negnevitsky. 2018. The Concept of the Deep Learning-Based System" Artificial Dispatcher" to Power System Control and Dispatch. *arXiv preprint arXiv:1805.05408* (2018).
[42] Caisheng Wang and M Hashem Nehrir. 2008. Power management of a stand-alone wind/photovoltaic/fuel cell energy system. *IEEE transactions on energy conversion* 23, 3 (2008), 957–967.
[43] xcelenergy.com. 2019. Time-of-Day pricing. Retrieved March 7, 2019 from https://www.xcelenergy.com/billing_and_payment/understanding_your_bill/residential_rate_plans/time_of_day
[44] Le Xie, Yingzhong Gu, Ali Eskandari, and Mehrdad Ehsani. 2012. Fast MPC-based coordination of wind power and battery energy storage systems. *Journal of Energy Engineering* 138, 2 (2012), 43–53.
[45] XI Yu-Geng, LI De-Wei, and Lin Shu. 2013. Model predictive control - status and challenges. *Acta Automatica Sinica* 39, 3 (2013), 222–236.
[46] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. 2016. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 528–535.