Integrated Energy Monitoring and Control IoT System and Validation Results from Neural Network Control Demonstration

Douglas Ellman, Pratiksha Shukla,
Yuanzhang Xiao, and Magdy Iskander
Hawai‘i Advanced Wireless Technologies Institute
and Department of Electrical Engineering
University of Hawai‘i at Mānoa
Honolulu, HI 96822 USA
Email: {dellman, shukla4, yxiao8, magdy}@hawaii.edu

Kevin Davies
Hawai‘i Natural Energy Institute
University of Hawai‘i at Mānoa
Honolulu, HI 96822 USA
Email: kdavies@hawaii.edu

***Abstract*—Increasing use of renewable and distributed power generation creates opportunities for customer resources to support power system operations by adjusting power consumption and generation to address grid needs, based on system-wide and local grid conditions. We present an integrated Energy Internet of Things (E-IoT) testbed including (1) distributed Advanced Real-time Grid Energy Monitor Systems (ARGEMS) with sensing, communication, and control capabilities, (2) distributed smart home sites, including smart home hubs for monitoring and control of physical and simulated Internet of Things (IoT) distributed energy resources (DERs) such as solar systems, home batteries, and smart appliances, and (3) control algorithms based on artificial intelligence and optimization, which manage customer DERs to respond to power grid conditions while serving customer needs. The integration of these three components enables demonstration and assessment of a variety of advanced DER monitoring and control strategies for improved power grid operations and customer benefits. We validate the functionality of this E-IoT testbed by demonstrating control of a simulated home battery by a neural network imitation learning algorithm running on a physical smart home hub, where the controller responds to grid services events triggered by an ARGEMS device based on local power system measurements and simulated bulk power system conditions. The developed neural network controller imitates the performance of a model predictive control optimization algorithm, but requires nearly 20,000 times less computational time and can run on small distributed computers.**

***Index Terms*—internet of things, smart grid, smart home, neural network, imitation learning, demand response**

I. Introduction

With the growing deployment of variable and distributed renewable generation sources, there are increasingly new use cases for customer and utility Internet of Things (IoT) devices to monitor and respond to electric grid conditions in order to maintain reliable operations and reduce system costs. For example, the grid integration of home generation (rooftop solar photovoltaic systems) and new high power devices such as electric vehicle chargers can create voltage and thermal issues at the distribution level, which can limit their adoption. The provision of bulk grid services from behind-the-meter (BTM) equipment including smart home appliances and batteries can either mitigate or exacerbate these issues, so it must be carefully coordinated especially with high penetrations of these distributed energy resources (DERs). The key is local grid awareness and control; therefore, there are benefits to systems where electricity prices and utility requests for bulk grid services can be filtered, arbitrated, or otherwise adjusted at the distribution level, even down to the service transformer, and calls for distribution grid services can be initiated locally. For these reasons, there is a compelling need for IoT systems that integrate distributed electric grid monitoring, control, and communications (e.g. at the level of distribution service transformers) with customers' DERs (such as controllable appliances, solar photovoltaics, battery systems, and electric vehicles), and can support advanced control schemes for a variety of grid service program structures.

IoT and other communication technologies have been used for decades to control customer resources for grid services in limited capacities. However, much work is still required to develop improved IoT devices, communications, and controls that enable coordinated use of large numbers of heterogeneous devices to flexibly respond to dynamic system-wide and local grid conditions under emerging power system markets, tariffs, and programs. Thus, there is a strong need for testbeds that can demonstrate and evaluate energy IoT systems in order to accelerate innovation and successful deployment.

Prior work on developing IoT architectures for customer energy resources include [1]–[3]. We build on this prior work, and present a new E-IoT testbed architecture that integrates three main components: (1) distributed Advanced Real-time Grid Energy Monitor Systems (ARGEMS) for electric grid monitoring, control, and communications (2) distributed smart home sites including smart home hubs and a variety of physical and simulated smart home IoT DER devices, and (3) advanced control algorithms based on artificial intelligence and optimization. The integration of these three components allows the E-IoT testbed to play a useful role in testing and

978-1-6654-3568-0/21/$31.00 ©2021 IEEE

optimizing the complete loop of grid monitoring to identify grid service needs, requesting grid services, and the response of customer resources. The E-IoT testbed supports development and testing of learning and optimization algorithms to manage DERs under a wide range of system-wide and local grid service applications, economic incentive structures, and communication and control topologies. The E-IoT testbed also incorporates low-cost and free and open source software in a way that is flexible (to support many existing and new IoT technologies) and scalable (supporting integration of hundreds or more physical and simulated sites and IoT devices).

We validate the E-IoT testbed's functionality by demonstrating advanced control of a battery system to minimize customer electric bills while responding to electric grid services requests. Due to the increasing deployment of battery systems (which are commonly installed with rooftop solar systems), and their capability to be quickly and accurately dispatched to generate or consume power, batteries can be powerful resources to support power system operations. While control algorithms based on solving large optimization problems can demonstrate the potential performance of battery systems under certain electricity tariffs and grid services programs [4], the significant computational resources required each time such a control algorithm is evaluated are an impediment to their widespread use for home battery systems. This motivates the development of alternative control algorithms that can also achieve high performance, but can be deployed and implemented with much less computational resources.

Based on this, and building on prior works [4]–[7], we developed a neural network controller that is trained to imitate a model predictive control (MPC) algorithm [4] but requires nearly 20,000 times less computational time per evaluation. We deployed and tested the neural network on a smart home hub managing a simulated battery system in the E-IoT testbed, where it responds to grid services events sent by a physical ARGEMS grid sensor and communication device. Important features of this neural network control demonstration include:

- integration of (1) an electric grid IoT device (ARGEMS), (2) a physical customer smart home hub and simulated IoT devices (home battery system, solar, and load sensor), and (3) an advanced neural network control algorithm in a realistic real-time demonstration of the E-IoT testbed;
- applying battery control domain knowledge to the neural network design to achieve high performance with a network small enough to run on smart home hubs; and
- design of the network, training loss function, and post-processing of outputs to respect battery physical constraints while maintaining high performance.

II. E-IoT Testbed Architecture

The E-IoT testbed includes distributed grid monitoring systems, distributed smart home sites, and central computers, which together enable a variety of distributed and centralized monitoring and control approaches.

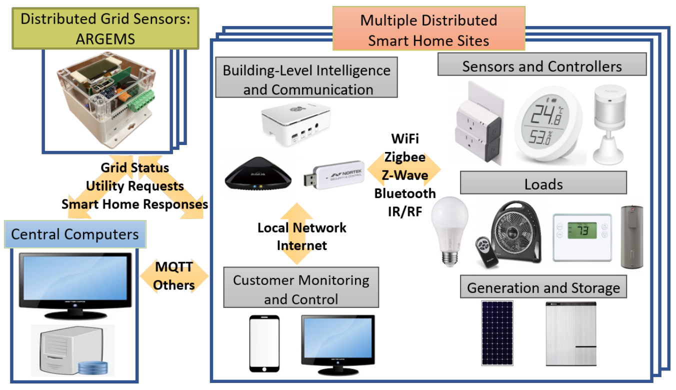

Fig. 1. Architecture of the E-IoT testbed.

A. Electric Grid Monitoring, Communications, and Control

In order for customer resources to dynamically support electric grid needs, we require: (1) information about the status of the bulk power system and local power distribution system, (2) the ability to process that information to identify needs for grid services, and (3) the ability to send grid services requests or economic signals to customers' devices. In the E-IoT testbed, these functions are provided by distributed grid sensors, central computers, or both, which enables a range of grid service applications at the bulk system and local level.

In this work, requests for grid services were initiated by a power monitor/controller called the Advanced Real-time Grid Energy Monitoring System (ARGEMS), which contains high-resolution multi-phase voltage and current data acquisition, a single board computer, power supply with backup, and various methods of communication (WiFi, Ethernet, 4G LTE, and wireless mesh). It also contains a timing-optimized GPS receiver and field programmable gate array (FPGA) for future applications in the realm of micro-synchrophasors (μPMU) and advanced, low-latency signal processing. The device has been custom-designed at the board level and is shown in Figure 1. The device is able to provide total and fundamental root mean square (RMS) voltages and currents, real and reactive power, grid frequency, as well as access to the underlying waveforms. This can be processed and published to web clients or upstream servers in near real-time or used for local purposes such as volt/VAR control or transformer health assessment. Research is ongoing to incorporate local RMS distribution power flow models to support online PV hosting capacity analysis and non-wires alternatives to manage voltage and thermal constraints.

This work is a step towards an integrated, scalable DER aggregation platform with distribution system awareness. Ultimately the E-IoT central computers and smart home sites will receive information on the status of the electric grid and electricity prices in addition to requests for grid services under a variety of existing and proposed utility programs.

B. Smart Home Sites

Smart homes in the E-IoT testbed monitor and control IoT DER devices (electric loads, generation, and energy storage) in response to customer needs and grid signals. Using multiple distributed smart home sites can represent variations in customer attributes (energy needs, available technologies, physical locations) and can demonstrate scalability of energy IoT concepts. Each smart home includes a smart home hub and IoT devices.

1) Smart Home Hub: The smart home hub is a computer that includes hardware and software to communicate with customer IoT devices, grid sensors, and central computers over IoT protocols. The smart home hub includes software to support advanced automated controls of IoT devices, provide local user interfaces, and interact with simulated IoT devices.

We use Raspberry Pi computers for smart home hubs, since they are inexpensive, compact, and have sufficient computing resources and input/output connections. The Raspberry Pi comes with support for WiFi and Bluetooth protocols, and we add a USB device to enable communication via Zigbee and Z-Wave protocols. These four IoT protocols enable connections to a large share of available smart home IoT devices. Hardware to support additional established or under-development IoT protocols can be added via USB ports or general-purpose input/output pins. As an alternative to physical computers, virtual machines can also serve as E-IoT smart home hubs.

The following free and open-source software are used by the E-IoT smart home hubs:

- Home Assistant smart home software supports integration of over 1,000 commercial smart home IoT devices over multiple protocols, allows custom integration of additional IoT devices, enables simple automation, and provides local web interface and mobile application,
- Mosquitto MQTT broker enables communication with MQTT IoT devices, with the E-IoT central computers, and with grid monitoring systems;
- Node-RED automation software enables the smart home but to automatically take actions based on time, receipt of MQTT messages, or other triggers, thus enabling interaction with other E-IoT computers, use of advanced control algorithms, and running IoT device simulations;
- PyTorch machine learning software supports training and deployment of machine learning control algorithms.

2) IoT Devices: The following categories of IoT devices can be managed by E-IoT smart home hubs:

- energy consuming and producing IoT devices, including lighting, air conditioning, fans, water heating, solar and battery systems, and electric vehicle chargers;
- sensors, including environment, occupancy, and power;
- smart plugs and switches, enabling on-off control and power monitoring of additional devices; and
- universal remote devices, enabling control of additional appliances via IR and RF signals.

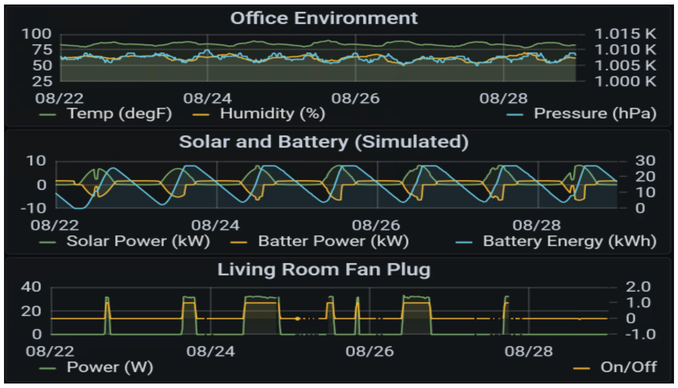

Fig. 2. Dashboard with integrated physical and simulated IoT device data.

C. Central Computers

The following "central" functions can be deployed on physical computers, virtual machines, or cloud servers:

- **database:** Node-RED reads MQTT messages with real-time smart home hub data and saves data to InfluxDB databases. Grafana dashboards display the data.
- **simulation:** U.S. Department of Energy's EnergyPlus software runs advanced physics-based building and device simulations based on weather conditions.
- **control:** Controls are implemented in Python, including PyTorch for machine-learning based control and Pyomo for optimization-based control. Central control decisions are communicated to the smart home hubs via MQTT.
- **training:** Data from the database and other sources are used to train and test control algorithms.

III. E-IoT Demonstration: Battery Control via Neural Network Imitation Learning

This section discusses the application of an imitation learning [8], [9] approach to develop and deploy a neural network that controls a home battery to minimize customer energy bills while responding to electric grid services events. The goal is to achieve a control algorithm that has high performance (i.e., achieves low customer bills) and that also has a low computational burden so that it can be deployed at scale on small distributed computers, such as smart home hubs.

We use a neural network to identify control actions, because neural networks have the ability to evaluate complicated functions with much less computational resources than it takes to solve a large optimization problem. In principle, we could train a neural network to directly learn control actions that minimize customer cost based on real-world or simulated experience (i.e., reinforcement learning). However, successful application of reinforcement learning to this problem would likely require a large amount of training time and data, because the customer cost impact of the current action generally depends on all of the future actions, with some costs and payments occurring infrequently (e.g., monthly grid services payments). We thus instead use an imitation learning approach to train the neural network, which reduces the reinforcement learning problem (learn optimal actions from experience interacting with the environment) to a supervised learning problem (learn the mapping from input to output in labeled data).

In our imitation learning approach, we train the network to imitate the performance of an optimization-based MPC algorithm [4]. We use the MPC algorithm to generate many examples of model inputs and control action outputs, and use those examples as training data for the neural network.

This section discusses the generation of training and testing data from the MPC algorithm, the process of designing, training, and testing neural networks to achieve high performance, modifications to the neural network designs to enforce battery physical constraints while maintaining high performance, and deployment of the neural network on a smart home hub in the E-IoT testbed.

A. Battery Control Problem Details

For a detailed mathematical description of the battery control problem for customer participating in baseline-based demand response programs, see our prior work [4]. Here, we summarize key details of the customer, tariff, and grid services programs considered in this study.

We consider a simulated Honolulu residential customer with typical electricity demand, a solar photovoltaic system, and a battery energy storage system. The customer electricity demand is the base-case Honolulu residential customer demand obtained from an OpenEI dataset [10], which includes hourly load profiles generated by the EnergyPlus building energy simulation for a typical year's weather based on the Typical Meteorological Year 3 (TMY3) dataset [11]. The solar production is obtained from PVWatts [12], based on a 7.56 kW rooftop system with default parameters and the Honolulu TMY3 weather data. Using TMY3 data for both customer demand and solar aligns their weather-based variations. The customer's battery simulates two units of the Tesla Powerwall battery [13], a popular residential battery system. The round-trip efficiency is 90%, and we assume equal charging and discharging efficiency of $\sqrt{0.9}$. The total battery system power capacity is 10 kW and energy capacity is 27 kWh. The battery and solar system size were selected based on a separate sizing analysis.

The customer participates in the Smart Export electricity tariff for residential customers with solar systems, which pays customers \$0.1497/kWh for energy exported to the grid from 4 p.m. to 9 a.m., pays nothing (\$0/kWh) for energy exports from 9 a.m. to 4 p.m., and charges the normal retail rate for energy consumed from the grid. We set the rate for energy purchases from the grid to \$0.29/kWh.

The customer participates in grid services that are based on the Capacity Reduction and Capacity Build grid services [14].[1] Under Capacity Reduction, the customer receives a DR capacity payment of \$2/kW for average load reduction during DR events each month, where DR events occur from 5 p.m. to 9 p.m. on event days. Load reduction is measured as the difference between baseline load (average load at 5 p.m. to 9 p.m on 10 prior non-event days) and load during the DR event. Under Capacity Build, the customer receives a DR capacity payment of \$3/kW for average load increase during DR events each month, where DR events occur from 10 a.m. to 2 p.m. on event days. Load increase is measured as the difference between load during the DR event and baseline load (average load at 10 a.m. to 2 p.m. on 10 prior non-event days).

Customers are notified of Capacity Reduction and Capacity Build either the day before the event (day-ahead) or on the day of the event (same-day). The utility calls events based on forecasted needs for load reduction or load build. Since the event calls are correlated with weather conditions, the customer's control algorithm can consider estimated probabilities of future events based on forecasted weather conditions. We assume the customer's controller has access to estimated event probabilities based on an analysis of historical events and weather conditions.

B. Training and Testing Data

The model predictive control algorithm [4] was used to generate data to train, validate, and test the neural network. The data was generated by simulating 13 sample passes through a "typical meteorological year," where each pass may vary due to random realizations of the event schedule and random sampling within the MPC algorithm.

Each data example, consisting of a vector of input values and output values, is generated from the corresponding input and output values of the MPC algorithm for each day of each sample year. Thus, there were 4,745 examples generated, corresponding to 365 days in 13 simulated years. 9 years of data was used for training (3,204 examples), 3 years for cross validation (1,068 examples), and 1 year for testing (365 examples).

Each data example has 1,254 elements in the input vector, corresponding to hourly solar and load forecasts for 25 days, daily Capacity Build and Capacity Reduction grid service event probability forecasts for 25 days, the current state of charge of the battery, the number of days remaining in the current month, and the number of prior Capacity Build and Capacity Reduction grid service events in the current month. The output vector has 24 values, corresponding to the amount to charge or discharge the battery in each hour of the upcoming day. The output values are scaled to a fraction of the maximum power capacity of the battery system, P.

MPC model parameters were held constant in the generated data, and thus are not included as elements in the neural network training, validation, and testing examples. These parameters include the customer battery system technical specifications (size, efficiency), the solar system energy generation, the default customer load, and the cost and payment rates for the customer electricity tariff and demand response programs.

[1] In the actual Capacity Reduction and Capacity Build programs, the customer receives flat payments that are not directly based on performance, while the customer's aggregator receives variable payments based on performance in the events. For simplicity, we consider in this paper that the customer directly receives variable performance-based payments.

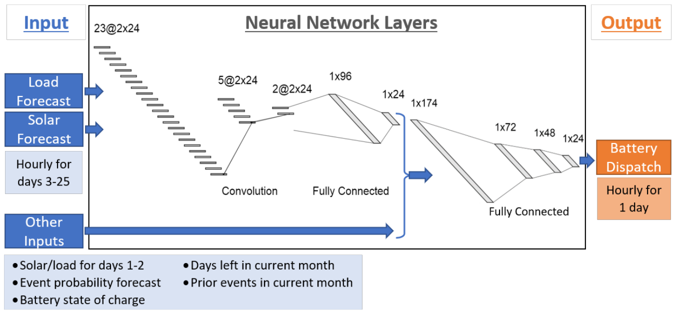

Fig. 3. Neural network inputs, outputs, and layers.

TABLE I
NEURAL NETWORK DESIGN

Convolutional Network		
Layer 1	Input	2x24x23 (day 3-25 hourly solar/load forecast)
	Layer type	2D conv., 1x1 filter, stride of 1, 5 output chan.
	Activation	rectified linear unit
	Output	2x24x5
Layer 2	Input	2x24x5
	Layer type	2D conv., 1x1 filter, stride of 1, 2 output chan.
	Activation	rectified linear unit
	Output	2x24x2, flattened to 1x96
Layer 3	Input	1x96
	Layer type	fully connected
	Activation	rectified linear unit
	Output	1x24
Fully Connected Network		
Layer 4	Input	1x174 (layer 3 output with remaining inputs)
	Layer type	fully connected
	Activation	rectified linear unit
	Output	1x72
Layer 5	Input	1x72
	Layer type	fully connected
	Activation	rectified linear unit
	Output	1x48
Layer 6	Input	1x48
	Layer type	fully connected
	Activation	hyperbolic tangent
	Output	1x24 (hourly battery dispatch for one day)

C. *Neural Network Design and Performance Evaluation*

This section discusses the process of designing a neural network that mimics the performance of the MPC algorithm. The design process was implemented in two stages. In the first stage, several neural network design options were tested in order to find a design that achieved customer costs close to the MPC algorithm, and significantly better than a simple heuristic algorithm. In this stage, respecting the battery's physical constraints of maximum and minimum stored energy was ignored. In the second stage, modifications to the neural network loss function and a post-processing step were developed in order to respect the battery constraints while still achieving high performance.

1) Neural Network Design: See Figure 3 and Table I for a summary of the neural network inputs, outputs, and design.

A key part of the neural network design concept is to split the input vector into two parts. The first part, consisting of 1,104 values corresponding to hourly load and solar forecasts for days 3 through 25, goes through a convolutional neural network that outputs a smaller number of features. Using a convolutional neural network to process this part of the solar and load forecasts allows the network to have a smaller number of trainable parameters compared to using all fully connected layers. Then, the 24 extracted features from the convolutional neural network are concatenated with the remaining 150 values from the input vector, and go through several fully-connected neural network layers. Note that based on our domain knowledge that the near-term forecast is more significant than the forecast for later days, we pass input values for the days 1 and 2 solar and load forecast deeper into the network, which saves the network from having to learn that the near-term forecast is more important. Having these near-term solar and load forecast values bypass the convolutional layers was demonstrated to significantly improve performance in testing (performance assessment is discussed in following sections).

The number and size of hidden layers in the network were selected by starting with a smaller network, and increasing the size until the network demonstrated satisfactory performance.

Rectified linear units are used as the activation functions in all layers except for the final layer. Rectified linear units introduce non-linearity into the neural network, which allows the network to represent more complicated functions. The final layer uses a hyperbolic tangent activation, which has a range of $(-1,1)$. Hyperbolic tangent was selected because its range aligns well with the range of the target output values, which is $[-1,1]$, where -1 corresponds to charging at the maximum power rating of the battery system, and 1 corresponds to discharging at the maximum power rating of the battery. This design ensures that the neural network outputs respect the physical constraint that the charging and discharging rates should always be less than or equal to the maximum power rating of the battery system.

2) Neural Network Training: In order to test the design of the neural network, network parameters were trained and selected using training and cross validation data sets, and then the trained neural network was compared with other reference control strategies based on customer cost in a test year.

The L1 loss was used for the imitation learning training loss function, $l_i(\mathbf{t},\mathbf{o})$. This is calculated as the sum of the absolute value of the differences between the target hourly battery dispatch, t_h, (from the training example) and the output hourly battery dispatch calculated by the neural network, o_h:

$$l_i(\mathbf{t},\mathbf{o}) = \sum_{h=1}^{24} |t_h - o_h|. \tag{1}$$

When selecting a loss function, we considered that the battery dispatch determined by the MPC algorithm is not the unique dispatch that achieves a certain cost, and there may be certain hours with large charging or discharging values that could be reallocated to other hours without a change to customer cost. The L1 loss function was selected instead of L2 loss (sum of squared deviations), because the L1 loss encourages minimizing average deviations, while the L2 loss would put more emphasis on reducing the large deviations on those hours with large charging or discharging values.

Training was performed by iterating through 3,000 epochs of the training data set using the L1 loss function, as described above, and the Adam optimizer, with parameters of: learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1×10^{-8}.

After every 50 epochs, the L1 loss of the neural network on the cross validation set was calculated, and the weights with the lowest cross validation loss were selected as the final weights.

3) Testing Neural Network Performance: The performance of the neural network was evaluated based on the customer cost achieved by using the neural network to control the battery in a simulated test year. We use customer cost as the performance metric rather than the imitation learning training loss function (total deviations in battery dispatch compared to the MPC algorithm), because minimizing customer cost is the objective for the MPC algorithm and is the ultimate goal of the imitation learning neural network as well.

In order to assess the quality of the annual customer cost achieved by the neural network, we compare it with the annual customer cost achieved by alternative control algorithms. We compare with three reference control algorithms: (1) heuristic controller that ignores grid services, (2) heuristic controller that maximizes response to grid service events, and (3) the MPC controller. Algorithm 1 tells the battery to charge as much as possible up to the level of solar generation minus home load (within the physical constraints of the battery) whenever solar production exceeds home load, and to discharge as much as possible up to the level of home load minus solar generation whenever home load exceeds solar generation. This algorithm aims to reduce the amount of energy that is exported to the grid, which lowers customer costs when the rate paid for exports is significantly lower than the rate charged for energy purchases. Algorithm 2 is the same as algorithm 1 outside the hours of an active grid services event. During grid services events, the battery operates to maximize the response to the event. In other words, the battery maximizes discharging within the battery physical constraints during Capacity Reduction events and it maximizes charging during Capacity Build events. Algorithm 3 follows the instructions of the MPC algorithm, which solves an optimization problem each day designed to minimize annual customer costs. Table II shows the cost of these three reference controllers, where algorithm 3 (MPC) achieves the lowest cost, followed by algorithm 1 (heuristic ignoring DR). Algorithm 2 (heuristic that maximizes DR) has the worst cost, which is likely because naively maximizing DR increases costs of energy purchases from the grid more than it increases DR payments in this example.

A good imitation learning network should achieve a cost that is similar to algorithm 3 (the target controller). It should also perform significantly better than algorithms 1 and 2 (simple heuristics), since there would be no point in using a neural network controller if a simple heuristic could provide comparable or better performance. Table II shows that the test year cost achieved with the imitation learning neural network ($1,074), was close to the cost achieved by the MPC algorithm ($1,060), and significantly better than the costs of the heuristic algorithms ($1,137 and $1,206).

TABLE II
COST PERFORMANCE OF REFERENCE CONTROLLERS AND NEURAL NETWORK.

Controller	Customer cost in test year
Heuristic - Ignore DR	$1,137
Heuristic - Max DR	$1,206
MPC	$1,060
Neural network	$1,074

4) Design Iteration for Respecting Constraints: While following a battery dispatch based on the neural network described above would provide good cost performance, this dispatch is actually not feasible because it violates physical constraints of the battery. Simulated battery dispatch based on the neural network sometimes causes the battery to store less than 0% or more than 100% of its energy capacity (27 kWh), which is not physically realizable. Thus, we implemented changes in neural network training and post-processing of outputs to ensure that physical constraints are respected.

The first change we implemented was to impose an additional term to the training loss function that imposes a loss when the battery energy constraints are violated. The new training loss, l, was calculated as:

$$l(\mathbf{t},\mathbf{o}) = l_i(\mathbf{t},\mathbf{o}) + l_c(\mathbf{o}), \tag{2}$$

where l_i is the previously used L1 imitation loss, and l_c is the new constraint violation loss. l_c is calculated as a scaled sum of the quantity of battery energy constraint violations according to the following equations, where p_h^+ and p_h^- are hourly battery charging and discharging values, P is the battery maximum power rating, e_h is the energy stored in the battery at the end of hour h, E is the battery energy storage capacity, η^+ and η^- are battery charging and discharging efficiencies, v_h is the hourly quantity of constraint violation, and α is a model parameter used to tune the relative importance of l_i and l_c:

$$p_h^+ = \begin{cases} -P \cdot o_h & \text{if } o_h < 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

$$p_h^- = \begin{cases} P \cdot o_h & \text{if } o_h > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

$$e_h = e_{h-1} + p_h^+ \eta^+ - p_h^- / \eta^-, \tag{5}$$

$$v_h = \begin{cases} -e_h & \text{if } e_h < 0, \\ e_h - E & \text{if } e_h > E, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

$$l_c = \frac{\alpha}{E} \sum_{h=1}^{24} v_h. \tag{7}$$

TABLE III
NEURAL NETWORK PERFORMANCE WITH LOSS FUNCTION PENALTY FOR CONSTRAINT VIOLATIONS.

Energy constraint penalty factor	Customer cost in test year	Lowest stored energy below min (kWh)	Highest stored energy above max (kWh)
$\alpha = 0$	\$1,074	-5.04	11.83
$\alpha = 10$	\$1,078	-1.42	2.69
$\alpha = 100$	\$1,094	0	1.16
$\alpha = 1,000$	\$1,253	0	0.58

Table III shows test cost and constraint violations resulting from training the neural network design with loss functions including constraint violation costs with α values of 0 (i.e., no penalty for constraint violation), 10, 100, and 1,000. As expected, with increasing values of α the amount of constraint violations decrease, but the customer cost increases. Ideally, we would have found a level of α such that constraint violations are eliminated but the customer cost performance is still good. However, as α increases to 1,000 the customer cost becomes significantly worse than the target MPC cost and the reference heuristic cost and there are still constraint violations.

Since using a training penalty for constraint violations was inadequate to achieve good customer cost with a physically realizable battery dispatch, we tried a complementary strategy where the final battery charging, p_h^+, and discharging, p_h^-, levels are determined by post-processing the neural network output, $\mathbf{o}$, as follows to ensure no constraint violations:

$$p_h^+ = \begin{cases} \min\{-P \cdot o_h,\ (E - e_{h-1})/\eta^+\} & \text{if } o_h < 0, \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

$$p_h^- = \begin{cases} \min\{P \cdot o_h,\ (e_{h-1})\eta^-\} & \text{if } o_h > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

$$e_h = e_{h-1} + p_h^+ \eta^+ - p_h^- / \eta^- . \tag{10}$$

Table IV shows the results of applying this post-processing procedure to the networks trained with α values of 0 (i.e., no constraint penalty in training), 10, and 100. In all of these cases, there are no constraint violations and the customer cost is significantly better than the heuristic reference cost. Interestingly, the best cost performance was achieved with a small but positive value of $\alpha = 10$. This indicates that there can be a benefit to first training the network with a small penalty for constraint violations (encouraging the network to learn to avoid violating constraints in ways with small impacts to cost performance) and then subsequently performing post-processing to ensure no constraint violations.

5) Performance Evaluation: For the final selected neural network trained with $\alpha = 10$ and with post-processing, we compared the performance of the neural network to the MPC algorithm. Figure 4 visually shows the neural network's ability to closely match the complicated dispatch pattern of the MPC algorithm. Table V compares the cost performance and evaluation time of the MPC algorithm neural network. Cost performance is shown as a cost reduction compared to the better performing heuristic control algorithm (the heuristic that ignores DR) in the test year. Evaluation time is the average time per evaluation when the algorithm is used 365 times for the test year. The results show that the neural network retains over 70% of the MPC's cost reduction from the heuristic control, and each neural network evaluation is nearly 20,000 times faster than the MPC algorithm.

TABLE IV
NEURAL NETWORK PERFORMANCE WITH POST-PROCESSING TO ENFORCE CONSTRAINTS.

Energy constraint penalty factor	Customer cost in test year
$\alpha = 0$	\$1,095
$\alpha = 10$	\$1,082
$\alpha = 100$	\$1,095

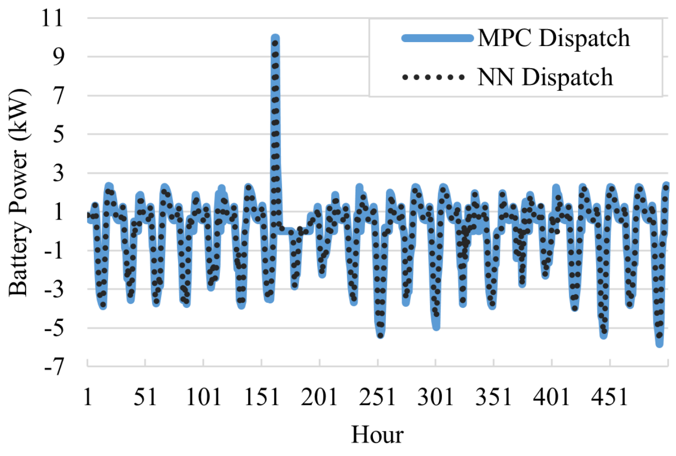

Fig. 4. Battery dispatch from the MPC and NN controllers for the first 500 hours of a test year.

D. Neural Network Deployment in E-IoT Testbed

The trained neural network was deployed in a real-time demonstration in the E-IoT testbed using a mix of physical hardware and communiation and simulated devices.

Node-RED software running on a smart home hub simulated real-time operations of the home's solar system, the home electric load, and a home battery at a 1-minute time granularity, which are integrated into the hub's Home Assistant smart home software as MQTT sensors. Each minute, Node-

TABLE V
NEURAL NETWORK AND MPC PERFORMANCE COMPARISON

Control Algorithm	Cost Reduction from Heuristic	Evaluation Time
MPC	6.8%	78 seconds
Neural Network	4.8%	4×10^{-3} seconds

RED reads and reports the home energy consumption based on EnergyPlus simulation and solar consumption based on PVWatts simulation. The battery simulation progresses each minute by reading the current target battery power level (which was determined by the neural network controller), and updating the battery's state of charge based on charging or discharging over the last minute, and reporting the current power and energy levels to Home Assistant. Home Assistant displays all of these devices within its user interface (along with other physical smart home devices), and reports data to the central E-IoT database compter.

An ARGEMS device was used to send notifications of Capacity Reduction and Capacity Build events via MQTT messages. Day-ahead notifications were sent at 9 p.m. each day for reduction and build grid service events corresponding to simulated bulk power system conditions (e.g. high total system load or high system level renewable generation). Additionally, same-day notifications were sent at 9 a.m. each day for Capacity Build events based on local grid conditions, where the event trigger was based on actual measurements of local solar production. This shows the capability for the testbed to demonstrate multiple grid services applications for bulk system and local grid conditions.

The smart home hub's Node-RED was also used to implement the battery scheduling algorithm, which compiles all of the input information for the neural network, calls a script to evaluate the neural network, and saves the schedule of target battery charging and discharging levels that is used in the battery simulation. In this demonstration, the neural network ran to update the battery schedule every 30 minutes, or whenever a new grid service event notification was received.

While this real-time demonstration has not yet operated long enough to validate the performance of the neural network in reducing customer costs, it has served as a validation of the ability to successfully deploy this neural network control with local and bulk system grid service event on a smart home hub without impacting the hub's ability to perform its other functions such as managing various IoT devices, running smart home software, and reporting data to the central database.

IV. Conclusion

We have presented an Energy Internet of Things (E-IoT) testbed that provides a flexible, scalable, and low-cost platform to demonstrate and evaluate the integration of (1) distributed ARGEMS power monitor/controllers, (2) distributed smart homes with IoT energy devices, and (3) advanced control algorithms that manage customer devices to respond to power grid conditions while serving customer needs. In the E-IoT testbed, ARGEMS devices monitor grid conditions (voltage, current, real and reactive power, and frequency), analyze needs for grid services, and transmit grid service requests and economic signals to the smart home sites. The smart home sites include smart home hubs that automatically monitor and control customer IoT energy devices to achieve customer objectives (e.g., reducing energy costs) while serving grid needs. Control algorithms based on artificial intelligence and optimization can be deployed on distributed smart home hubs, distributed ARGEMS devices, or centralized computers to demonstrate a variety of bulk power system and local grid services applications and control strategies.

We demonstrated and validated the E-IoT testbed's capabilities by developing and deploying an imitation learning neural network controller for a simulated home battery that responds to grid services signals from an ARGEMS device based on local power system measurements and simulated bulk power system conditions. We showed that the neural network imitates the performance of a computationally intensive MPC algorithm (successfully lowering customer costs and respecting battery physical constraints), but it requires nearly 20,000 times less computational time than the MPC algorithm and can successfully run on smart home hub computers.

The E-IoT testbed can support additional future research areas, including developing novel strategies for managing customer resources for grid service applications, as well as evaluating performance and security of energy IoT systems.

References

[1] J. Pan, R. Jain, S. Paul, T. Vu, A. Saifullah, and M. Sha, "An internet of things framework for smart energy in buildings: Designs, prototype, and experiments," *IEEE Internet Things J.*, vol. 2, no. 6, pp. 527–537, 2015.

[2] Y. Wang, L. Hou, K. Su, A. Miao, B. Chen, and Z. Zhang, "Research on integrated energy management and control system architecture based on local energy internet," in *2019 IEEE 3rd Conf. EI2*, pp. 6–11.

[3] H. Chaouch, A. S. Bayraktar, and C. Çeken, "Energy management in smart buildings by using m2m communication," in *2019 7th Int. ICSG*, pp. 31–35.

[4] D. Ellman and Y. Xiao, "Model predictive control-based battery scheduling and incentives to manipulate demand response baselines," 2019, arXiv:1909.12349.

[5] T. Li, Y. Xiao, and L. Song, "Deep reinforcement learning based residential demand side management with edge computing," in *2019 IEEE Int. Conf. SmartGridComm*, pp. 1–6.

[6] Y. Xiao and M. van der Schaar, "Foresighted demand side management," *IEEE Trans. Smart Grid (Forthcoming)*.

[7] D. Ellman and Y. Xiao, "Incentives to manipulate demand response baselines with uncertain event schedules," *IEEE Trans. Smart Grid*, vol. 12, no. 2, pp. 1358–1369, 2021.

[8] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[9] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[10] Office of Energy Efficiency & Renewable Energy (EERE), "Commercial and Residential Hourly Load Profiles for all TMY3 Locations in the United States." [Online]. Available: https://openei.org/doe-opendata/dataset/commercial-and-residential-hourly-load-profiles-for-all-tmy3-locations-in-the-united-states

[11] National Renewable Energy Laboratory, "National Solar Radiation Data Base: 1991- 2005 Update: Typical Meteorological Year 3." [Online]. Available: https://rredc.nrel.gov/solar/old_data/nsrdb/1991-2005/tmy3/

[12] ——, "PVWatts Calculator." [Online]. Available: https://pvwatts.nrel.gov/pvwatts.php

[13] Tesla, "Powerwall — The Tesla Home Battery." [Online]. Available: https://www.tesla.com/powerwall

[14] Hawaiian Electric, "Addendum No. 2 To Request for Proposals For Provision of Grid Services Utilizing Demand-Side Resources," 2018. [Online]. Available: https://www.hawaiianelectric.com/documents/products_and_services/demand_response/dr_rfp_best_and_final_offer.pdf