Direct Load Control of Thermostatically Controlled Loads Based on Sparse Observations Using Deep Reinforcement Learning

Frederik Ruelens, Bert J. Claessens, Peter Vrancx, Fred Spiessens, and Geert Deconinck

***Abstract*—This paper considers a demand response agent that must find a near-optimal sequence of decisions based on sparse observations of its environment. Extracting a relevant set of features from these observations is a challenging task and may require substantial domain knowledge. One way to tackle this problem is to store sequences of past observations and actions in the state vector, making it high dimensional, and apply techniques from deep learning. This paper investigates the capabilities of different deep learning techniques, such as convolutional neural networks and recurrent neural networks, to extract relevant features for finding near-optimal policies for a residential heating system and electric water heater that are hindered by sparse observations. Our simulation results indicate that in this specific scenario, feeding sequences of time-series to an Long Short-Term Memory (LSTM) network, which is a specific type of recurrent neural network, achieved a higher performance than stacking these time-series in the input of a convolutional neural network or deep neural network.**

***Index Terms*—Convolutional networks, deep reinforcement learning, long short-term memory, residential demand response.**

I. Introduction

OPTIMAL control of Thermostatically Controlled Loads (TCLs), such as heat pumps and water heaters, is expected to play a key role in the application of residential demand response [1]–[3]. TCLs can use their thermal inertia, e.g. a water buffer or building envelope, as a thermal battery to store energy and shift energy consumption in response to changes in the electricity price or to provide grid services. Among the more important challenges hindering the application of residential demand response is partial observability of the environment [4]–[6], where a part of the state remains hidden from the agent due to sensor limitations, resulting in a partially observed control problem.

Model-predictive Control (MPC) [7] and Reinforcement Learning (RL) [8] are two opposing paradigms to solve the optimal control problem of TCLs. As such, MPC and RL have developed a set of different techniques to tackle the problem of planning under partial observability.

In MPC, a Kalman filter is often used to estimate hidden features by exploiting information about the system dynamics and using Bayesian interference. For example, in [4] Vrettos *et al.* applied a Kalman filter to estimate the temperature of a building envelope and in [5] Kazmi *et al.* applied a similar approach to estimate the state of charge of an electric water heater.

RL approaches, on the other hand, store sequences of past interactions with their environment in a memory and extract relevant features based on this memory. The challenges herein is to consider a priori how many interactions are important to learn a specific task and what exact features should be extracted. Deep neural networks or multi-layer perceptrons are the quintessential technique for automatic feature extraction in RL [9], [10]. An important breakthrough of automatic feature extraction using deep learning is presented in [11], where Mnih *et al.* apply a convolutional neural network to automatically extract relevant features based on visual input data to successfully play Atari games.

Finally, by combining RL and MPC, the authors of [12] presented a method that trains complex control policies with supervised learning, using MPC to generate the supervision. The teacher (MPC) uses a rough representation of its environment and full state, and the learner updates its policy based on the partial state using supervised learning.

II. Literature Review

This section provides a short literature overview of Reinforcement Learning (RL) related to demand response and discusses some relevant applications of deep learning in RL.

A. Reinforcement Learning and Demand Response

An important challenge in tackling residential Demand Response (DR) is that any prior knowledge in the form of a physical model of the environment and disturbances is not readily available or may be too costly to obtain compared to the financial gains obtained with DR. As RL techniques can be applied "blind" and consider their environment as a black box, they require no prior knowledge nor do they require a system identification step, making them extremely suited for residential DR. As a result, residential DR has become

Manuscript received March 18, 2019; revised August 24, 2019; accepted October 17, 2019. Date of publication December 30, 2019; date of current version November 4, 2019.

F. Ruelens was and G. Deconinck (corresponding author, e-mail: geert.deconinck@kuleuven.be) is with the Department of Electrical Engineering, KU Leuven/EnergyVille, 3001 Leuven, Belgium.

B. J. Claessens is with REstore, Centrica, 2600 Antwerp, Belgium.

P. Vrancx was with the AI-lab, Vrije Universiteit Brussel, 1050 Brussels, Belgium.

F. Spiessens is with Vito/EnergyVille, 2600 Mol, Belgium.

DOI: 10.17775/CSEEJPES.2019.00590

a promising application domain for RL [13]–[19]. The most important RL algorithms applied to DR are temporal difference RL, batch RL and more recently deep RL. The first application of RL to demand response were standard temporal difference methods, such as Q-learning and SARSA [8]. For example, in [13], Wen *et al.* showed how Q-learning can be applied to a residential demand response setting and in [14], Kara *et al.* applied Q-learning to provide short-term ancillary services to the power grid by using a cluster TCLs. Extending this work, Mocanu *et al.* demonstrated how a deep belief network can be integrated in Q-learning and SARSA to extract relevant features [15], allowing for cross-building transfer learning.

In [16], the authors demonstrated how batch RL can be tailored to a residential demand response setting using a set of *hand-crafted* features, based on domain specific insights. The authors extended a well-known batch RL algorithm, fitted Q-iteration, to include a forecast of the exogenous variables and demonstrated that it outperformed standard temporal difference methods, resulting in a learning phase of approximately 20-30 days, suggesting that batch RL techniques are more suitable for demand response.

More recently, inspired by advances in deep learning, the authors extended this approach for a cluster of TCLs using an automatic feature extraction method based on convolutional neural networks [17]. A binning algorithm is used to map the full state of the cluster to a two-dimension representation that can be used as input for the convolution neural network. Similarly, in [18], François-Lavet *et al.* applied a convolutional neural network as a function approximator within RL to capture the stochastic behavior of the load and renewable energy production in a microgrid setting with a short-term and long-term storage. Finally, an exhaustive review and analysis of RL applied to electric power systems decision and control problems can be found in [19]. They state that due to the complexity of the electric power system, RL methods provide effective solutions to tackle electric power system control and decision.

B. Recurrent Neural Networks and Partial Observability

In contrast to vanilla neural networks, Recurrent Neural Networks (RNNs) have an internal state, which is based on the current input state and the previous internal state, allowing the internal state to act as a memory modeling the impact of previous input states on the current task. This internal state allows the RNN to process sequences of input data, making it a natural framework to mitigate the problem of partial state information.

In practice, however, RNNs have difficulties in learning long-term dependencies [20]. An LSTM network is a special type RNN developed by Hochreiter and Schmidhuber in [21] that solves the long-term dependency problem, by adding special structures called *gates* that regulate the flow of information to the memory state.

The application of a RNN within Q-learning, called recurrent-Q, was introduced by Lin and Mitchell in [22], demonstrating that recurrent-Q was able to learn non-Markovian tasks. Extending on this idea, Bram Bakker [23] demonstrated how LSTM using advantage learning can solve non-Markovian tasks with long-term temporal dependencies. In addition to value-based RL, a successful implementation of a policy gradient method with an LSTM architecture to a non-Markovian task can be found in [24]. Motivated by the promising results of Deepmind with Deep QN [11], the authors of [25] demonstrated how an LSTM network can be combined with a deep Q-network for handling partial observability in Atari games, induced by flickering game screens.

C. Contributions

This paper investigates the effectiveness of different deep learning techniques within reinforcement learning for demand response applications that are hindered by sparse observations, making the following contributions. We present how an LSTM network, Convolutional Neural Network (CNN) and multi-layer neural network, can be used within a well-known batch RL algorithm, fitted Q-iteration, to approximate the Q-function, extending the state with historic partial observations. We demonstrate their performance for two popular embodiments of flexible loads, namely a heat pump for space heating and an electric water heater. We compare the performance of LSTM, CNN, neural networks and extremely randomized trees when using sparse state information. The paper is structured as follows. Section III states the problem and formalizes it as a Markov decision process. Section IV explains how these deep learning techniques can be used to extract relevant features based on sequences of observations and used within a batch RL. Section V describes the different deep learning architectures. Section VI presents the simulation results of two flexibility carriers and finally Section VII draws conclusions and discusses further work.

III. Markov Decision-making Formalism

This section states the problem and presents the formalism to tackle it.

A. Problem Statement

In most complex real-world problems, such as demand response, an agent cannot measure the exact full state of its environment, but only a partial observation of the state. Depending on how good this partial observation can be used to model future interactions, using partial information may result in sub-optimal policies. This paper presents two demand response applications that are hindered by partial observability, where the agent cannot measure the state directly, but has to extract relevant features based on how much energy the application consumed and how much it lost. In our first experiment, we consider a heat-pump agent that can only measure its electricity consumption and outside temperature. In the second experiment, we consider an electric water heater agent with partial state information, consisting of its measured electricity consumption and the flow rate and temperature of the tap water exiting the water buffer. In addition, due to the sequential nature of both problems, the agent must take future transitions into account, resulting in a complex *sequential* decision-making problem under uncertainty.

To tackle this challenge, we will first formalize the underlying problem as a Markov decision process and then introduce the concepts of partial state information.

B. Formalism

At each discrete time step k, the *full* state of the environment evolves as follows: $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, u_k, \boldsymbol{w}_k)$ with $\boldsymbol{w}_k$ a realization of a random disturbance drawn from a conditional probability distribution $p_\omega(\cdot)$ and $u_k \in U$ the control action. Associated with each action of the agent, a cost c_k is provided by $c_k = \rho(\boldsymbol{x}_k, u_k, \boldsymbol{w}_k)$, where ρ is a cost function that is a priori given.

The goal of the agent is to find an optimal control policy $h^* : X \to U$ that minimizes the expected T-stage return for any state in the state space. Value-based RL techniques characterize the policy h is by using a state-action value function or Q-function:

$$Q^h(\boldsymbol{x}, u) = \mathop{\mathbb{E}}_{\boldsymbol{w} \sim p_\omega(\cdot)} \left[\rho(\boldsymbol{x}, u, \boldsymbol{w}) + J_T^h(f(\boldsymbol{x}, u, \boldsymbol{w})) \right] \quad (1)$$

The Q-function is the cumulative return starting from state x, taking action u, and following h thereafter. Given the Q-function, an action for each state can be found as:

$$h(\boldsymbol{x}) = \arg\ \min Q^h(\boldsymbol{x}, u). \quad (2)$$

This paper applies a value-based batch RL technique to approximate the Q-function corresponding to the optimal policy based on an imperfect observation of the true state.

C. Partial State

It is assumed that the state space X measured by the agent consists of three components: timing-related state information X^{time}, controllable state information X^{phys}, and exogenous (uncontrollable) state information X^{exo}. In this work the timing related is given by the current quarter in the day $x_k^{\text{time}} \in X^{\text{time}} = \{1, \ldots, 96\}$, which allows the agent to capture time-varying dynamics. The controllable state information $\boldsymbol{x}^{\text{phys}} \in X^{\text{phys}}$ comprises the operational measurements that are influenced by the control action. In reality, most agents can only measure a partial observation $\boldsymbol{o}_k^{\text{phys}}$ of the true state $\boldsymbol{x}_k^{\text{phys}}$, resulting in a partially observable Markov decision problem. The exogenous information $\boldsymbol{x}_k^{\text{exo}}$ is invariant for control actions u_k, but has an impact on the dynamics. Examples of exogenous variables are weather conditions and demand profiles (e.g heat demand).

Thus, the state measured by the agent at step k is given by:

$$\boldsymbol{x}_k^{\text{obs}} = (x_k^{\text{time}}, \boldsymbol{o}_k^{\text{phys}}, \boldsymbol{x}_k^{\text{exo}}). \quad (3)$$

Note that since (3) only includes part of the true state, it becomes impossible to model future state transitions, making the state non-Markovian.

D. Action

At each time step, a demand response agent can request an action $u_t \in [0, 1]$: either to switch OFF or ON. To guarantee the comfort and safety constraints of the end users, each TCL is equipped with an overrule mechanism (or thermostat). The backup function $B : X \times U \longrightarrow U^{\text{phys}}$ maps the requested control action $u_k \in U$ taken in state $\boldsymbol{x}_k$ to a physical control action $u_k^{\text{phys}} \in U^{\text{phys}}$:

$$u_k^{\text{phys}} = B(\boldsymbol{x}_k, u_k). \quad (4)$$

The settings of the backup function B are unknown to the learning agent, but the resulting action u_k^{phys} can be measured by the learning agent.

E. Cost

This papers considers a dynamic pricing scenario where an external price profile is known deterministically at the start of the optimization horizon:

$$c_k = \rho(u_k^{\text{phys}}, \lambda_k) = u_k^{\text{phys}} \lambda_k \Delta t \quad (5)$$

where λ_k is the electricity price at time step k and Δt is the length of a control period.

IV. Batch Reinforcement Learning

Given full observability, batch RL algorithms start with a batch of four tuples of the form: $\left(\boldsymbol{x}_k, u_k, \boldsymbol{x}'_k, u_k^{\text{phys}}\right)$, where $\boldsymbol{x}_k$ represents the true state of the problem.

According to the theory of partial observable Markov decision processes [9], the optimal value function at time step k depends on the partial state observations of *all* proceeding periods. However, since these observations accumulate over time, it is important to capture sufficient statistics, i.e. a history length h which summarizes the essential content of the measurements. As such, this paper tackles the problem of partial observability by augmenting the state vector with a sequence of partial state observation, requested actions and physical actions of the last h observations:

$$\boldsymbol{x}_k^{\text{aug}} = \left(x_k^{\text{time}}, \boldsymbol{x}_k^{\text{hist}}, \boldsymbol{x}_k^{\text{exo}}\right) \quad (6)$$

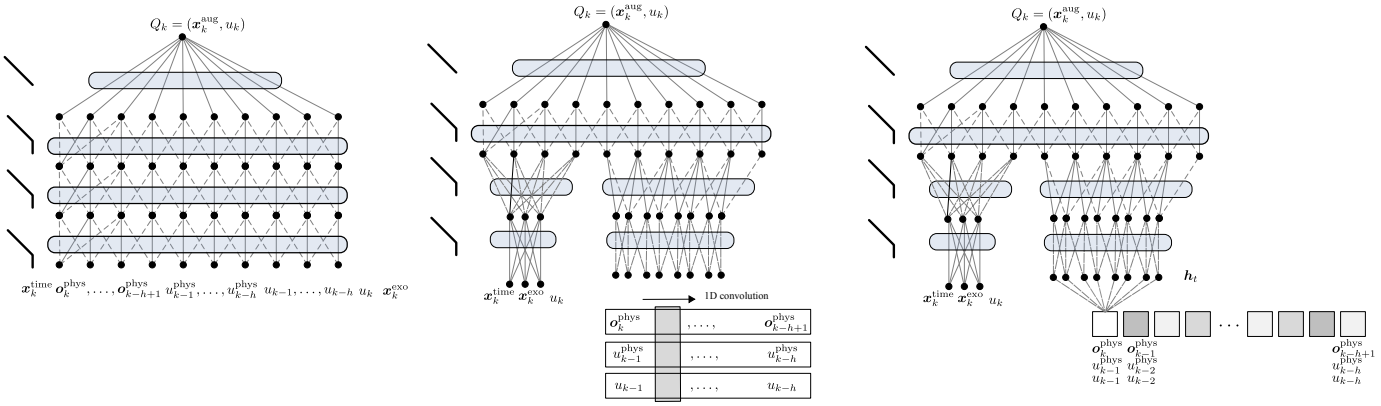
with $\boldsymbol{x}_k^{\text{hist}}$ given by:

$$[\boldsymbol{o}_k^{\text{phys}}, \ldots, \boldsymbol{o}_{k-h}^{\text{phys}}], [u_{k-1}^{\text{phys}}, \ldots, u_{k-1-h}^{\text{phys}}], [u_{k-1}, \ldots, u_{k-1-h}]. \quad (7)$$

As a result, this paper starts from a bath of four tuples given by: $\{(\boldsymbol{x}_k^{\text{aug}}, u_k, \boldsymbol{x}_k^{\text{aug}'}, u_k^{\text{phys}})\}_{k=1}^{\mathcal{M}}$, where $\boldsymbol{x}_k^{\text{aug}}$ represents the augmented state. An important challenge is to learn how to extract relevant features in a scalable way.

This paper applies fitted Q-iteration [26] to obtain an approximation of the Q-function $\widehat{Q^*}(\boldsymbol{x}^{\text{aug}}, u)$. Fitted Q-iteration iteratively approximates the Q-functions for each state-action pair using its corresponding cost and the approximation of the Q-function from the previous iterations. To leverage the availability of forecasts of exogenous information, e.g. outside temperatures, we use the extension of fitted Q-iteration as presented in [16], which replaces the observed exogenous information by its forecasted value $\hat{\boldsymbol{x}}_l^{\text{exo}'}$ (line 7 in Algorithm 1).

In order for Algorithm 1 to work, we need to select an approximator architecture (step 10) that is able to learn relevant features from sequences of input data and that can generalize the Q-function.



(a) Neural network (b) Convolutional neural network (c) Long short-term memory network

Fig. 1. Sketch of the deep learning architectures used in the simulation section. The LSTM network is represented as an unfolded computational graph, where each node is associated with one particular time instance.

Algorithm 1 Batch RL [26] using LSTM [21]

Input: $\mathcal{F} = \{\boldsymbol{x}_k^{\text{obs}}, u_k, \boldsymbol{x}_k^{\text{obs}'}, u_l^{\text{phys}}\}_{k=1}^{\#\mathcal{F}}$, $[\lambda_k]_{l=1}^{T}$, $[\hat{\boldsymbol{x}}_l^{\text{exo}}]_{l=1}^{T}$, history length h

construct $\mathcal{M} = \{\boldsymbol{x}_k^{\text{aug}}, u_k, \boldsymbol{x}_k^{\text{aug}'}, u_l^{\text{phys}}\}_{k=1}^{\#\mathcal{M}}$ using (6) and (7)

let $\widehat{Q}_0$ be zero everywhere on $X \times U$

for $N = 1, \ldots, T$ **do**

  for $k = 1, \ldots, \#\mathcal{M}$ **do**

    $l \leftarrow x_k^{\text{time}}$

    $c_k \leftarrow u_k^{\text{phys}} \lambda_l \Delta t$

    $\boldsymbol{x}_k^{\text{aug}'} \leftarrow (x_k^{\text{time}'}, \boldsymbol{x}_k^{\text{hist}'}, \hat{\boldsymbol{x}}_l^{\text{exo}'})$

    $Q_{N,k} \leftarrow c_k + \min_{u \in U} \widehat{Q}_{N-1}(\boldsymbol{x}_k^{\text{aug}'}, u)$

  end for

  use approximator (Fig. 1) to obtain $\widehat{Q}_N$ from $\mathcal{T} = \{((\boldsymbol{x}_k^{\text{aug}}, u_l), Q_{N,k}), k = 1, \ldots, \#\mathcal{M}\}$

end for

Output: $\widehat{Q}^* = \widehat{Q}_N$

V. Deep Learning Approximators

This paper investigates the effectiveness of the following deep learning approximators when combined with fitted Q-iteration.

A. Deep Neural Network

It has been shown that a neural network with a single layer is sufficient to represent any function, but the layer may become infeasible large and may fail to train and generalize correctly. To overcome these two challenges, deeper networks are used as these networks can reduce the number of units to represent the function and can reduce the generalization error. Fig. 1(a) illustrates the neural network as used in this paper, consisting of an input layer given by $(\boldsymbol{x}_k^{\text{aug}}, u_k)$, two hidden layers with rectified linear units (ReLUs), and one linear output layer, representing the approximated Q-function.

B. Convolutional Neural Network

CNNs have been successfully applied to extract features from image data, represented as a 2D grid of pixels. In this paper, we consider a time series and convolve a 1D filter of length N over the time-series in the state (7). A sketch of the applied CNN can be seen in Fig. 1(b), which consists of two components that are merged to output a singe value. The first component is a dense neural network which takes the timing-related information, exogenous information and action as input. The second component is a CNN which takes the time-series as input (7). For each sequence, the network consists of two layers containing eight 1D filters of length 4 followed by a ReLU, which is downsampled by using an average pooling layer.

C. Long Short-term Memory

1) Background

An LSTM network (Fig. 1(c)) consists of LSTM nodes that are recurrently connected to each other. Each LSTM node has an internal recurrence or memory cell $C^{(t)}$ and a system of gating units that controls the flow of information. For each step t of the sequence $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(h)}$, the resulting action of the forget gate $\boldsymbol{f}^{(t)}$, input gate $\boldsymbol{i}^{(t)}$ and output gate $\boldsymbol{o}^{(t)}$ of a single LSTM node is provided by:

$$\begin{aligned} \boldsymbol{f}^{(t)} &= \sigma(W_f[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}] + \boldsymbol{b}_f) \\ \boldsymbol{i}^{(t)} &= \sigma(W_i[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}] + \boldsymbol{b}_i) \\ \boldsymbol{o}^{(t)} &= \sigma(W_o[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}] + \boldsymbol{b}_o) \end{aligned} \tag{8}$$

where W_f, W_i, W_o and $\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_o$ are the weights and biases of the forget, input and output gate, σ denotes the logistic sigmoid function and $\boldsymbol{x}^{(t)}$ denotes the current element of sequence (11), with the time step index t ranging from 1 to h.

The internal memory cell of the LSTM node is updated as:

$$\boldsymbol{C}^{(t)} = \boldsymbol{f}^{(t)} * \boldsymbol{C}^{(t-1)} + \boldsymbol{i}^{(t)} * \widetilde{\boldsymbol{C}}^{(t)} \tag{9}$$

where $\widetilde{\boldsymbol{C}}^{(t)}$ and $\boldsymbol{C}^{(t-1)}$ are the current and previous memory state and $*$ denotes a pointwise multiplication operator. Note that the new memory $\boldsymbol{C}^{(t)}$ is defined by the information it forgets from the old state $\boldsymbol{f}^{(t)} * \boldsymbol{C}^{(t-1)}$ and remembers from the current $\boldsymbol{i}^{(t)} * \widetilde{\boldsymbol{C}}^{(t)}$.

In the last step, a hyperbolic tangent function is applied to the memory cell and multiplied with the output $\boldsymbol{o}^{(t)}$, which

defines what information to output.

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} * \tanh(\boldsymbol{C}^{(t)}) \tag{10}$$

This gating mechanism allows the LSTM network to store information about the state for long periods of time and protects the gradient in the cell from harmful changes during training related to the vanishing or exploding gradient problem of RNN [20].

2) Approximator Architecture

The approximator architecture consists of two components: an LSTM network and a standard multi-layer perceptron (Fig. 1). The first part of the input, corresponding to the LSTM component, contains the historic information of the partial state $\boldsymbol{x}^{\text{hist}}$. For each $k = h+1, \ldots, \#\mathcal{M}$, the input of the LSTM network is given by the following sequence:

$$\underbrace{\begin{bmatrix} \boldsymbol{o}_k^{\text{phys}} \\ u_{k-1}^{\text{phys}} \\ u_{k-1} \end{bmatrix}}_{\boldsymbol{x}_k^{(1)}}, \ldots, \underbrace{\begin{bmatrix} \boldsymbol{o}_{k-h+1}^{\text{phys}} \\ u_{k-h}^{\text{phys}} \\ u_{k-h} \end{bmatrix}}_{\boldsymbol{x}_k^{(h)}} \tag{11}$$

The history depth h defines how much time steps the network can see in the past to compute its approximation of the Q-function. The length of the memory cell d_{cell} represents an important hyper parameter and defines how many knowledges can be encoded. As can be see in Fig. 1 only the content of the last memory cell h_t is used as an input for the next layer.

The second part contains the time-related information, exogenous information and action: $x_k^{\text{time}}, \boldsymbol{x}_k^{\text{exo}}, u$. The outputs of both components are combined to form a single architecture, which is followed by two fully connect layers with Rectified Linear Unit (ReLU) activation functions. A final linear output layer approximates the final Q-function for the provided state-action pair.

VI. Simulation Experiments

This section evaluates the performance of combining the presented deep learning techniques with Algorithm 1 for two TCLs (a heat-pump within building and an electric boiler) as providers of demand flexibility exposed to a dynamic energy price. The TCLs are represented using a second-order equivalent thermal parameter (ETP) model.

A. Simulation Framework

At the end of each simulation day, Algorithm 1 is used to compute a new policy based on current batch and electricity price for the following day. The RL agent starts with an empty batch and alternates exploration and exploitation according to a decreasing exploration probability: $\varepsilon_d = 0.75^d$, where d denotes the current episode.

All experiments are repeated 10 times starting from a different random seed, resulting in different exploration probabilities and stochastic disturbances. The results below present the average of these simulation experiments, where a confidence bound ($\pm 2\sigma$) is indicated by a shaded area, representing a 0.95 probability that the solution lies in the shaded area.

The average simulation time for one day (Algorithm 1) is about 1.5 hour[1] using Keras with Theano as backend.

B. Experiment 1: Space Heating

The second-order ETP model for the space heating considers both the inside air temperature as well as the (not observable) building mass temperature, similar as in [17], [27]. The dynamics of the model are given by

$$\begin{aligned} \frac{dT_{\text{a}}}{dt} &= \frac{1}{C_{\text{a}}} \left[T_m H_m - T_{\text{a}} \left(U_{\text{a}} + H_m \right) + Q_{\text{a}} + T_o U_{\text{a}} \right] \\ \frac{dT_m}{dt} &= \frac{1}{C_m} \left[H_m \left(T_{\text{a}} - T_m \right) + Q_m \right] \end{aligned} \tag{12}$$

where U_{a} equals the conductance of the building envelope. T_{o} is the outside air temperature, T_{a} is the inside air temperature, and T_{m} is the inner mass temperature. H_{m} is the conductance between the inner air and the solid mass. C_{a} and C_{m} represent the thermal mass of the air and interior solid mass, respectively. The heat flux into the interior air mass Q_{a} is given by a combination of solar heat gains, heat gains of the internal loads, and heat gain generated by the heat pump itself, which is related to the heat pump power multiplied with its coefficient-of-performance, as in [28].

Numerically, a second-order heat-pump model ($C_a = 2.441$ MJ/K, $U_a = 125$ W/K, $C_m = 9$ MJ/K, $H_m = 6.863$ kW/K) with real-world Belgian outside temperatures T_{o} from [29] is used to simulate the temperature dynamics of a residential building with a heat pump. The heat pump has a maximum electric heating power of 2.3 kW and the minimum and maximum comfort settings are set to 20°C and 23°C (293 K and 296 K). To model stochastic impact of user-behavior we sample an exogenous temperature disturbance from $\mathcal{N}(0, 0.025)$. The time resolution of the dynamics is 60 seconds and of the control policy is 15 minutes.

The state vector describing the environment is defined as:

$$\boldsymbol{x}_k = (x_k^{\text{time}}, T_k^{\text{a}}, T_k^{\text{m}}, T_k^{\text{o}}, T_k^{\text{exo}}), \tag{13}$$

where x_k^{time} contains timing information, T_k^{a} the air temperature, T_k^{m} the virtual mass temperature, T_k^{o} the outside temperature and T_k^{exo} an exogenous disturbance. As stated in the problem description, it is assumed that the RL agent cannot measure the air and mass temperature of the building, resulting in a partial observed control problem. As such, we construct the following augmented state vector:

$$\begin{aligned} \boldsymbol{x}_k^{\text{aug}} = \big(& x_k^{\text{time}}, [u_{k-1}^{\text{phys}}, \ldots, u_{k-h}^{\text{phys}}], \\ & [u_{k-1}, \ldots, u_{k-h}], \\ & [T_{k-1}^{\text{o}}, \ldots, T_{k-h}^{\text{o}}], T_k^{\text{o}} \big), \end{aligned} \tag{14}$$

which includes three time series of length $h = 20$.

1) NN Architecture

The neural network consists of three dense layers with 50 neurons with ReLU activation functions, followed by a linear output unit. The neural network was trained using RMSprop with a minibatch size of 32, and the training set contained about 100 days of data at 15 minute resolution (hence, approximately 10,000 data points).

[1]Simulation hardware: Xeon E5-2680 v2 processor with 15 GiB memory (Amazon elastic cloud instance type: c3.2xlarge).

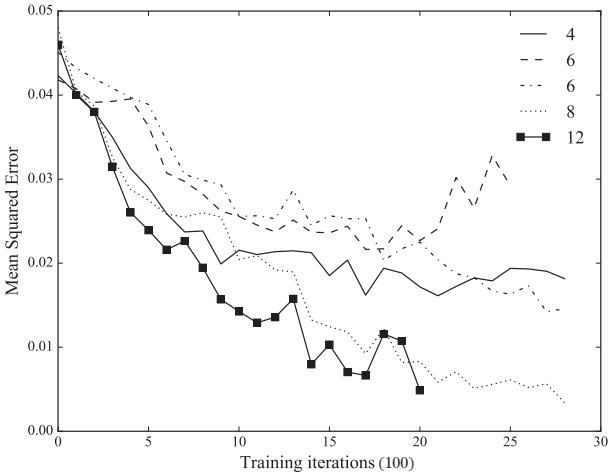


Fig. 2. Mean squared error during training for different dimensions of the LSTM memory cell. The training was stopped after 10 minutes. The best results obtained after 10 minutes of training using a memory cell of dimension 8.

2) CNN Architecture

The network consists of two components, namely a CNN and dense network. The CNN component consists of two 1D convolutions (along the time dimension) that are each followed by an average pooling layer. The dimension of the first filter is $(L \times 3)$, where L is the filter length and 3 is number of input sequences and the dimension of the second filter is $L \times 1$. Both filters have a filter length of 4. The dense network processes the time-related information, exogenous information and action. Both components are merged and followed with two layers with 20 neurons and a single output layer. All layers use ReLU activation function except for the output layer that uses a linear function.

3) LSTM Architecture

The input to the LSTM network is provided by the sequence:

$$\begin{bmatrix} u_{k-1}^{\text{phys}}, \\ u_{k-1}, \\ T_{k-1}^{\text{o}} \end{bmatrix}, \ldots, \begin{bmatrix} u_{k-h}^{\text{phys}}, \\ u_{k-h}, \\ T_{k-h}^{\text{o}} \end{bmatrix} \tag{15}$$

and the NN is provided by $x_k^{\text{time}}, T_k^{\text{o}}$ and u_k.

For the heat-pump experiment the best results were obtained with the history depth h set to 20 time steps (quarters) and the length of each LSTM memory cell d_{cell} set to 8.

4) Convergence

Figure 3 depicts the cumulative cost using function approximators (Fig. 3(a)) and daily average outside temperature (Fig. 3(b)). The no control strategy activates the backup controller, without setting a control action, and can be seen as a worst case scenario as it is agnostic about the electricity price. An upper bound is computed by considering the full state information as defined in (13). In addition to LSTM with partial state information, the figure depicts the cumulative of using an ensemble of extremely randomized trees (or ExtraTrees) [26]. The number of trees in the ensemble was set to 100 and the minimum sample size for splitting a node to 5. Our results indicate that the ExtraTrees approximator was not able to extract relevant features from the partial state information and performed only 1.5% better than the no control strategy. In contrast, the LSTM approximator was able to extract relevant features and achieved a reduction of 5.5%.

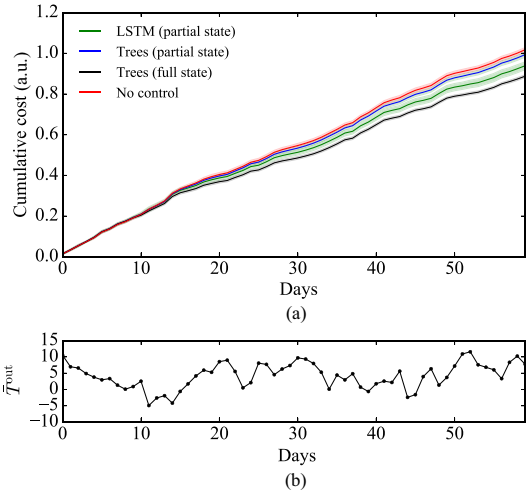


Fig. 3. (a) Cumulative cost of the heat pump experiment using FQI-LSTM and FQI-Trees. (b) Corresponding daily average outside temperature.

Figure 4(a) shows the daily cost obtained with Algorithm 1, using a partial state information and a neural network, with CNN and LSTM network as a function approximator. Fig. 4(b) indicates the scaled cost which is c calculated as follows: $(c - c_{\text{f}})/(c_{\text{nc}} - c_{\text{f}})$, where c_{f} is the result of using the full state information and c_{nc} of using the no control strategy, resulting in $c = 0$ for the full state strategy and $c = 1$ for the no control strategy. This figure indicates Algorithm 1 obtained a scaled cost of 0.37 using LSTM, 0.66 using NN and 0.82 using CNN. Fig. 4(c) compares the resulting control policies of LSTM, CNN en NN with the control policy of the

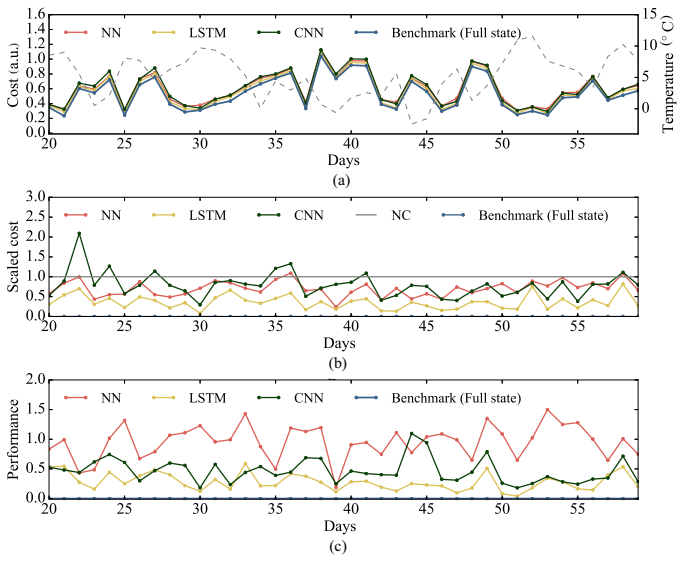


Fig. 4. (a) Daily cost for the heat pump experiment using FQI-NN, FQILSTM and FQI-CNN based on sparse observations. (b) Corresponding scaled daily cost. (c) Metric defined by the distance between the near optimal policy (benchmark) and policy obtained with FQI-NN, FQI-LSTM and FQI-CNN.

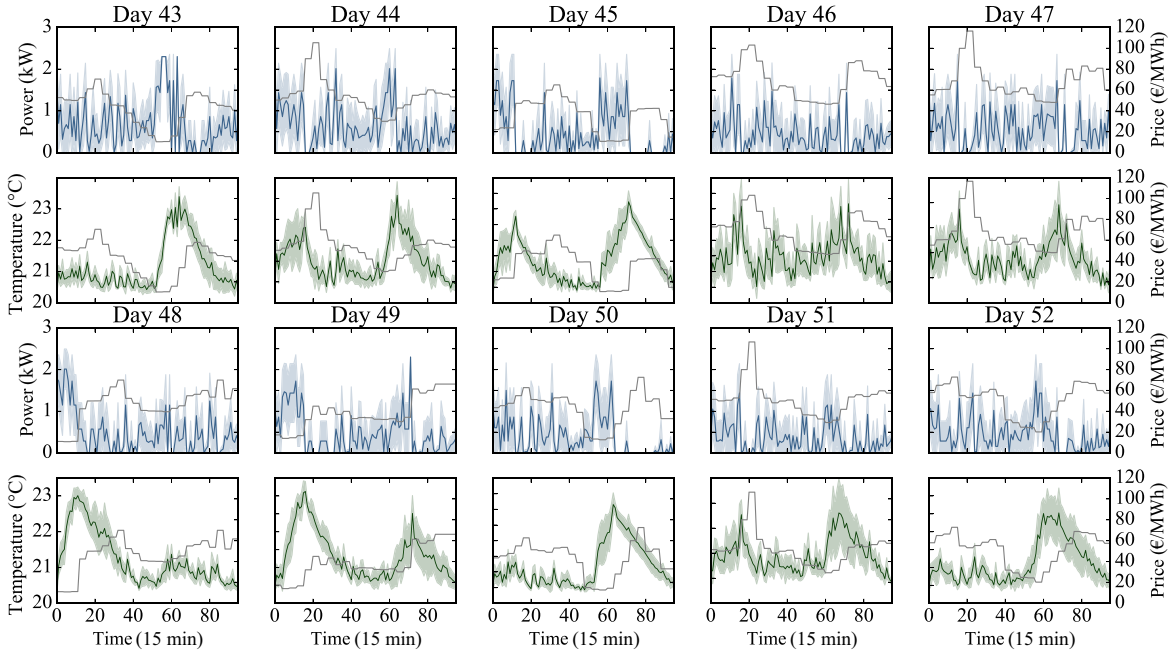


Fig. 5. Power consumption (first and third row) and air temperatures (second and fourth row) for 10 greedy simulation days (left y-axis) using FQI-LSTM with partial state information for the heat pump experiment. The corresponding price profiles are depicted in gray (right y-axis).

full state using a euclidean distance. Although NN achieved a better performance than CNN, the resulting policy of CNN and LSTM are closer to the policy of the full state. We speculate that the CNN and LSTM learned a better representation of the full state than the NN, since the NN achieved a low cost by lowering the air temperature to minimum temperature without reacting to the price.

5) Daily Results

A more qualitative interpretation of our results can be seen in Fig. 5. The figure shows the power consumption and the corresponding daily price profiles. It can be seen that the learning agent successfully postponed its power consumption to low price moments, while satisfying the comfort constraints.

C. Experiment 2: Electric Water Heating

The second experiment considers an electric water heater with a water buffer of 200 liters and a daily average water consumption of 100 liter. The minimum and maximum water temperature is set to 45°C and 65°C. The water heater is equipped with a thermostatic mixing value to assure a constant requested temperature of 45°C. The water heater has an electric power rating of 2.3 kW and a built-in backup controller as defined in [30]. The time resolution for the dynamics is 5 seconds and the time resolution for the control policy is 15 minutes.

The full state vector of the electric water heater is defined by:

$$\boldsymbol{x}_k = \left(x_k^{\text{time}}, T_k^1, \ldots, T_k^{|\mathcal{L}|}, d_k\right) \tag{16}$$

where $T_{1,t}^i$ is the temperature corresponding to the i^{th} layer and d_k is the current tap demand. During our simulation, a non-linear stratified model with 50 layers is used to simulate the temperature gradient along the water tank and stochastic tap water profiles are used based on [30].

In a previous paper [31], the authors considered that the agent could measure an imperfect state through eight temperature sensors. In this experiment, however, it is assumed that the buffer is not equipped with a set of sensors to measure the different temperatures inside the water buffer.

As a result, we define the following augmented state vector:

$$\boldsymbol{x}_k^{\text{aug}} = \big(x_k^{\text{time}}, [u_{k-1}, \ldots u_{k-h}], [u_{k-1}^{\text{phys}}, \ldots u_{k-h}^{\text{phys}}] \\ [\dot{m}_k, \ldots \dot{m}_{k-h+1}], [T_k^{|\mathcal{L}|}, \ldots, T_{k-h+1}^{|\mathcal{L}|}]\big) \tag{17}$$

where x_k^{time} contains timing information, u_k is the requested control action, u^{phys} is the actual action, and $\dot{m}_k$ and $T_k^{|\mathcal{L}|}$ are the mass flow rate and temperature of the water exiting the water buffer. Note that $[u_{k-1}^{\text{phys}}, \ldots u_{k-h}^{\text{phys}}]$ represents the electricity consumption of the boiler and $[\dot{m}_k, \ldots \dot{m}_{k-h+1}], [T_k^{|\mathcal{L}|}, \ldots, T_{k-h+1}^{|\mathcal{L}|}]$ represents the energy flowing out of the boiler.

1) (C)NN Architecture

The NN and CNN architecture are identical as in the previous experiment with the exception that the filters size of the first convolutional layer is 4×4, because now we have 4 input sequences.

2) LSTM Architecture

The input to the LSTM network is provided by the sequence:

$$\begin{bmatrix} u_{k-1}^{\text{phys}}, \\ u_{k-1}, \\ \dot{m}_k, \\ T_k^{|\mathcal{L}|} \end{bmatrix}, \ldots, \begin{bmatrix} u_{k-h}^{\text{phys}}, \\ u_{k-h} \\ \dot{m}_{k-h+1}, \\ T_{k-h+1}^{|\mathcal{L}|} \end{bmatrix} \tag{18}$$

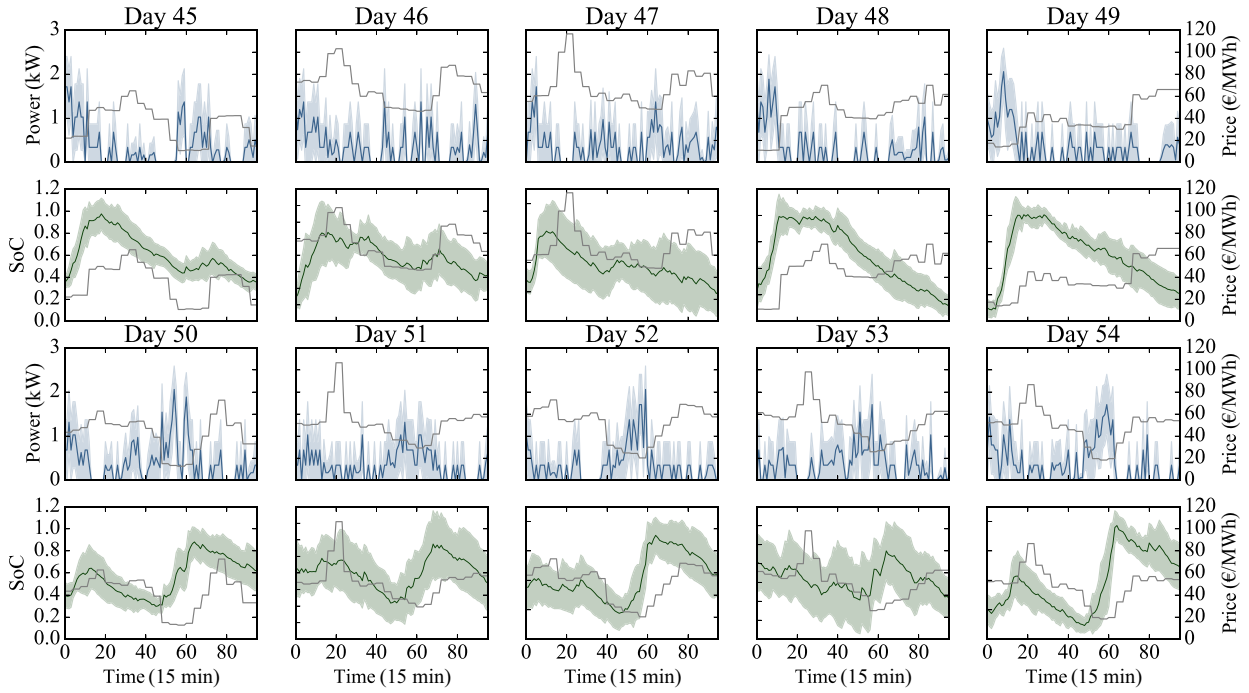


Fig. 6. Power consumption (first and third row) and state of charge (second and fourth row) for 10 greedy simulation days (left y-axis) using FQI-LSTM with partial state information for the electric water heater experiment. The corresponding price profiles are depicted in gray (right y-axis).

For the boiler experiment the best results were obtained with the history depth h set to 40 time steps (quarters) and the length of each LSTM memory cell d_{cell} set to 12.

3) Daily Results

For the electric water heater scenario, we only offer qualitative results (Fig. 6). It shows the daily power consumption of an electric water heater and corresponding price profiles. It can be seen that the learning agent required four weeks of learning before obtaining reasonable policies (lower row of graphs). A final comparison between using a CNN or LSTM network as a function approximator can be seen in Fig. 7, indicating that using a CNN resulted in a cost reduction of 5.5% and using an LSTM network in 10.2%. The results of FQI-NN were omitted because we were able to stabilize the learning of the NN.

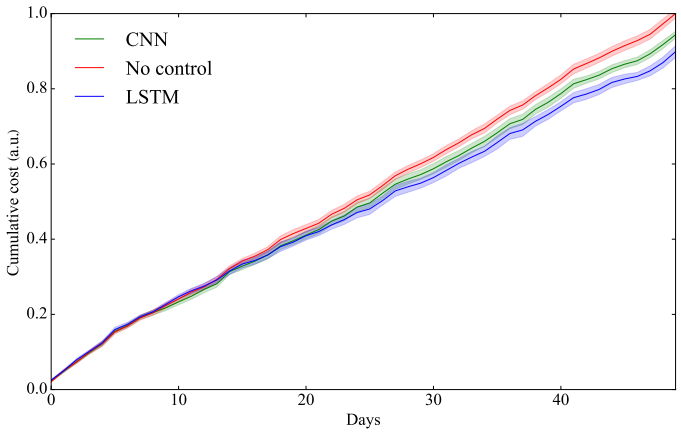


Fig. 7. Cumulative cost of the electric water heater experiment using FQI-LSTM and FQI-CNN based on partial state information.

VII. Conclusion

In this paper, we demonstrated the effectiveness of combining different deep learning techniques with reinforcement learning for two demand response applications that are hindered by sparse observations of the true state. Since these sparse observations result in a non-Markovian control problem, we extended the state with sequences of past observations of the state and action.

In the first experiment, we considered an agent that controls a residential heating system under a dynamic pricing scenario, where the agent can only measure its electricity consumption, control action and outside temperature. Our simulations indicated that reinforcement learning with long short-term memory (LSTM) performed better than other techniques such as a neural network, convolutional neural network and ensemble of regression trees, when sparse observations were used. In our second experiment, we considered an agent that controls a residential electric water heater with a hot storage vessel of 200 liter. In this scenario, the agent can only measure its electricity consumption, control action and flow and temperature of the tap water exiting the storage vessel. The simulation results indicated that the LSTM network outperformed the convolutional network and deep neural network.

We speculate that the higher performance of the LSTM network comes from its internal memory cell which can act as an integrator. This internal memory cell allows the LSTM network to process sequences of sparse observations and extract relevant features from it that can represent the underlying state of charge (or energy level) of the application.

References

[1] J. L. Mathieu, M. Kamgarpour, J. Lygeros, G. Andersson, and D.S.

Callaway, "Arbitraging intraday wholesale energy market prices with aggregations of thermostatic loads," *IEEE Transactions on Power Systems*, vol. 30, no. 2, pp. 763–772, Mar. 2015.

[2] B. Dupont, P. Vingerhoets, P. Tant, K. Vanthournout, W. Cardinaels, T. De Rybel, E. Peeters, and R. Belmans, "LINEAR breakthrough project: Large-scale implementation of smart grid technologies in distribution grids," in *Proceedings of the 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, Berlin, Germany, 2012, pp. 1–8.

[3] G. Deconinck and K. Thoelen, "Lessons from 10 years of demand response research: Smart energy for customers?" IEEE Systems, Man, and Cybernetics Magazine, vol. 5, no. 3, pp. 21–30, Jul. 2019.

[4] E. Vrettos, E. C. Kara, J. MacDonald, G. Andersson, and D. S. Callaway, "Experimental demonstration of frequency regulation by commercial buildings—part I: Modeling and hierarchical control design," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3213–3223, Jul. 2018.

[5] H. Kazmi, S. D'Oca, C. Delmastro, S. Lodeweyckx, and S. P. Corgnati, "Generalizable occupant-driven optimization model for domestic hot water production in NZEB," *Applied Energy*, vol. 175, pp. 1–15, Aug. 2016.

[6] Q. Hu, F. Oldewurtel, M. Balandat, E. Vrettos, D. T. Zhou, and C. J. Tomlin, "Building model identification during regular operation - empirical results and challenges," in *Proceedings of 2016 American Control Conference (ACC)*, 2016, pp. 605–610.

[7] E. F. Camacho and C. Bordons, *Model Predictive Control*, 2nd ed., London, UK: Springer London, 2004.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning*: *An Introduction*, Cambridge, MA: MIT Press, 1998.

[9] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Nashua, NH: Athena Scientific, 1996.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*[Online]. MIT Press, 2016. Available: http://www.deeplearningbook.org

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[12] G. Kahn, T. Z. Zhang, S. Levine, and P. Abbeel, "PLATO: Policy learning using adaptive trajectory optimization," *in Proceedings of 2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3342–3349.

[13] Z. Wen, D. O'Neill, and H. Maei, "Optimal demand response using device-based reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2312–2324, Sep. 2015.

[14] E. C. Kara, M. Berges, B. Krogh, and S. Kar, "Using smart devices for system-level management and control in the smart grid: A reinforcement learning framework," in *Proceedings of the 3rd IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Tainan, China, 2012, pp. 85–90.

[15] E. Mocanu, P. H. Nguyen, W. L. Kling, and M. Gibescu, "Unsupervised energy prediction in a smart grid context using reinforcement cross-building transfer learning," *Energy and Buildings*, vol. 116, pp. 646–655, Mar. 2016.

[16] F. Ruelens, B. J. Claessens, S. Vandael, B. De Schutter, R. Babuška, and R. Belmans, "Residential demand response of thermostatically controlled loads using batch reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2149–2159, Sep. 2017.

[17] B. J. Claessens, P. Vrancx, and F. Ruelens, "Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3259–3269, Jul. 2018.

[18] V. François-Lavet, D. Taralla, D. Ernst, and R. Fonteneau, "Deep reinforcement learning solutions for energy microgrids management," in *European Workshop on Reinforcement Learning (EWRL 2016)*, 2016.

[19] M. Glavic, R. Fonteneau, and D. Ernst, "Reinforcement learning for electric power system decision and control: Past considerations and perspectives," in *Proceedings of the 20th World Congress of the International Federation of Automatic Control (IFAC), Toulouse 9–14 July*, Toulouse, France, 2017, pp. 1–10.

[20] Y. Bengio, P. Frasconi, and P. Simard, "The problem of learning long-term dependencies in recurrent networks," in *IEEE International Conference on Neural Networks*, 1993, pp. 1183–1188.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[22] L. J. Lin and T. M. Mitchell, "Reinforcement learning with hidden states," in *Proceedings of the 2nd international conference on From Animals to Animats 2: Simulation of Adaptive Behavior*, 1993, pp. 271–280.

[23] B. Bakker, "Reinforcement learning with long short-term memory," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, Vancouver, British Columbia, Canada, 2001, pp. 1475–1482.

[24] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber, "Solving deep memory POMDPs with recurrent policy gradients," in *Proceedings of the 17th International Conference on Artificial Neural Networks*, 2007, pp. 697–706.

[25] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," arXiv:1507.06527, 2015.

[26] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, pp. 503–556, Apr. 2005.

[27] W. Zhang, K. Kalsi, J. Fuller, M. Elizondo, and D. Chassin, "Aggregate model for heterogeneous thermostatically controlled loads with demand response," in *Proceedings of 2012 IEEE Power and Energy Society General Meeting*, 2012, pp. 1–8.

[28] S. Iacovella, F. Ruelens, P. Vingerhoets, B. Claessens, and G. Deconinck, "Cluster control of heterogeneous thermostatically controlled loads using tracer devices," *IEEE Transactions on Smart Grid*, vol. 8, no. 2, pp. 528–536, Mar. 2017.

[29] KMI - Royal Meteorological Institute of Belgium. (2017, January 21). Ambient temperatures (Ukkel, Belgium)[Online]. https://github.com/openideas/IDEAS/blob/master/IDEAS/Inputs/Uccle.TMY

[30] K. Vanthournout, R. D'hulst, D. Geysen, and G. Jacobs, "A smart domestic hot water buffer," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 2121–2127, Dec. 2012.

[31] F. Ruelens, B. J. Claessens, S. Quaiyum, B. De Schutter, R. Babuska, and R. Belmans, "Reinforcement learning applied to an electric water heater: From theory to practice," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3792–3800, Jul. 2018.

Frederik Ruelens received his Ph.D. degree in Electrical Engineering from the KU Leuven, Belgium in 2016. Since 2019, he is Deployment Engineer at Proximus focusing on DevOps (docker, kubernetes, automated continuous integration). He is passionate about solving complex problems from different fields and deploying real-world machine learning applications.

Bert J. Claessens received his Ph.D. degree in applied physics from the University of Technology of Eindhoven, The Netherlands in 2006. Since 2016 he is head of optimization research and innovation at REstore as part of Centrica Business Solutions. Since 2019 he is part-time professor of intelligent energy systems at Eindhoven University of Technology. His main research interests are directed towards residential demand response and artificial intelligence for energy applications.

Peter Vrancx obtained his Ph.D. degree in Computer Science summa cum laude from Vrije Universiteit Brussel. During his Ph.D. he developed game theoretical models of multi-agent reinforcement learning algorithms. After graduating, he worked as a postdoctoral researcher on several applied machine learning and reinforcement learning projects. In 2016, he became an Assistant Professor with the Department of Computer Science at the Vrije Universiteit Brussel. He also served as a steering group member of BruBotics - the Brussels Human Robotics Research Center. He is currently the Head of Reinforcement Learning and Multi-agent Systems research at PROWLER.io. His main research interests are Reinforcement Learning, Neural Networks, and Game Theory.

Alfred (Fred) Spiessens received his Ph.D. degree in Software Engineering from the Universit Catholique de Louvain, Belgium in 2007. He has experience in industry since 1983 and performs applied and academic research since 2004. Since 2014 he is a Senior Research Professor in the group Algorithms, Models and Optimization at EnergyVille/VITO investigating energy optimization techniques and advising Ph.D. students.

Geert Deconinck is Full Professor at KU Leuven university (Belgium). He received his M.Sc. degree in Electrical Engineering and his Ph.D. degree in Engineering Sciences from KU Leuven, Belgium in 1991 and 1996 respectively. He is head of the research group ELECTA on Electrical Energy at the Department of Electrical Engineering (ESAT). In the research centre EnergyVille on smart energy for sustainable cities, he is the scientific leader for the algorithms, modelling, optimisation, applied to smart electrical and thermal networks. His research focuses on robust distributed coordination and control, specifically in the context of smart grids.