Online End-to-End Learning-based Predictive Control for Microgrid Energy Management

Vittorio Casagrande, Martin Ferianc, Miguel Rodrigues, Francesca Boem

***Abstract*—This paper proposes an innovative Online Learning (OL) algorithm designed for efficient microgrid energy management, integrating Recurrent Neural Networks (RNNs) and Model Predictive Control (MPC) in an End-to-End (E2E) learning-based control architecture. The algorithm leverages the RNN capabilities to predict uncertain and possibly evolving profiles of electricity price, load demand, and renewable generation. These are then exploited in an integrated MPC optimisation problem to minimise the overall microgrid electricity consumption cost while guaranteeing operation constraints. The proposed methodology incorporates a specifically designed online version of the Stochastic Weight Averaging (O-SWA) and Experience Replay (ER) methods to enhance online learning capabilities, ensuring more robust and adaptive learning in real-time scenarios. Additionally, to address the challenge of model uncertainty, a task-based loss approach is proposed by integrating the MPC optimisation as a differentiable optimisation layer within the neural network, allowing the OL architecture to jointly optimise prediction and control performance. The performance of the proposed methodology is evaluated through extensive simulation results, showcasing its Transfer Learning (TL) capabilities across different microgrid sites, which are crucial for deployment in real microgrids. We finally show that our OL algorithm can be used to estimate the prediction uncertainty of the unknown profiles.**

***Index Terms*—Energy management system, Microgrid, Online learning, Model predictive control, End-to-end learning, Learning-based control**

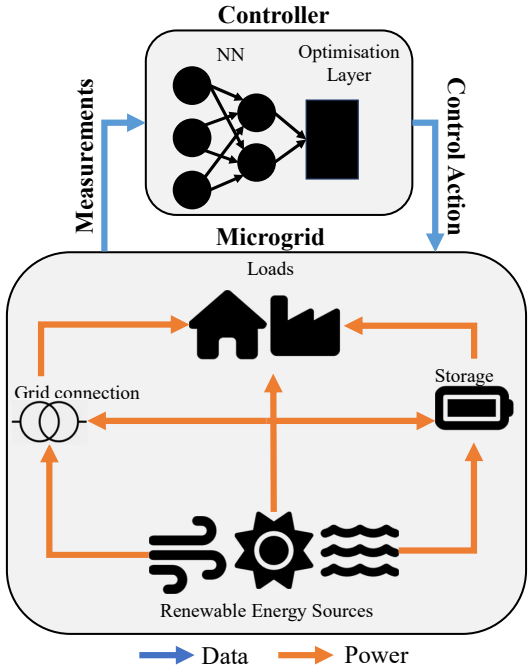

Fig. 1. Schematic of the control architecture. The controller receives current measurements of the system states and current values of the unknown profiles from the microgrid and then computes the power schedule.

NOMENCLATURE

List of Acronyms

EMS	Energy Management System
ER	Experience Replay
HPO	Hyper-parameters optimisation
ML	Machine Learning
MPC	Model Predictive Control
NN	Neural Network
O-SWA	Online Stochastic Weight Averaging
OL	Online Learning
RNN	Recurrent Neural Network
TL	Transfer Learning

Controller hyperparameters

α_v	Regression loss weight for the feature v
β	Task loss weight
ϵ	ER loss weight
γ, δ	O-SWA decay factor and update period
n_h	LSTM hidden dimension

List of symbols

$\psi_t^\bullet$, $\omega_t^\bullet$	NN input/prediction at time t
σ, η_{in}, η_{out}	Storage parameters
L	Look-back window
P_t^g	Power exchanged with the grid at time t
P_t^l	Load power at time t
P_t^r	Renewable power generation at time t
$P_t^{s_{in}}$, $P_t^{s_{out}}$	Storage input and output power at time t
p_t	Electricity price at time t
s_m, s_M, P_M^s	Storage charge and power limits
s_t	Storage state at time t
T	MPC prediction horizon
T_s	Controller sampling time

Vittorio Casagrande and Martin Ferianc contributed equally. This work has been supported by the Engineering and Physical Sciences Research Council (grant reference: EP/W024411/1). For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising. Vittorio Casagrande was sponsored through a scholarship from the Engineering and Physical Sciences Research Council (grant reference: EP/R513143/1). Martin Ferianc was sponsored through a scholarship from the Institute of Communications and Connected Systems at UCL. The authors are with the Department of Electronic and Electrical Engineering, University College London, London, United Kingdom {vittorio.casagrande.19,martin.ferianc.19, m.rodrigues,f.boem}@ucl.ac.uk

I. INTRODUCTION

THE microgrid concept, firstly introduced in [35], is defined in [44] as a cluster of loads, distributed generation units and energy storage systems, operated in coordination to reliably supply electricity and connected to the power distribution grid. The Microgrid higher control layer is responsible for power flow control, power management and optimisation of the microgrid operation [53] and includes the microgrid EMS. This is defined in [59] as the controller that computes the power flows to provide a stable delivery of power to loads while optimising energy production and other operational goals. One of the primary challenges in designing a microgrid EMS is effectively addressing the uncertainty, particularly concerning electricity price, load demand, and renewable power generation. The conventional approach to tackle this issue involves the separate design of a predictor to estimate the uncertain profiles and of an optimiser to compute the microgrid schedule accordingly. Typically, statistical tools are utilised as predictors based on historical data. In the literature, offline trained (i.e. before deployment) ML models are widely proposed for this purpose [18]. However, the first drawback of this approach is that such predictors can't adapt to changes in the environment, requiring retraining whenever significant changes occur, such as installing new equipment, revamping, maintenance, or the occurrence of faults. To address this limitation, OL techniques offer a solution by enabling the predictor to continuously adapt based on the incoming stream of data [22]. Moreover, another drawback of the separate design of the prediction and optimisation tools is that this might lead to sub-optimal control performance. Performance-based learning methods [33] are offering efficient solutions for problems involving prediction and optimisation, such as microgrid scheduling [15]. In this paper, we propose a novel method that integrates a comprehensive OL architecture, enabling the predictor's adaptability in real-time and MPC, allowing the computation of the optimal scheduling while guaranteeing the satisfaction of operational and safety constraints. Additionally, we design an online version of performance-based learning to further enhance the overall performance of the EMS by taking into account the uncertainty in the model adopted by the MPC (such as, for example, the uncertainty in the energy conversion efficiency of a storage system), so to optimise prediction and control performance jointly. A schematic of the proposed architecture is shown in Fig. 1. The controller comprises a NN part for profiles' prediction and an integrated optimisation layer for the scheduling computation. The controller receives the microgrid's measurements of the microgrid state and the current values of renewable generation, load demand and price profiles and computes the optimal power schedules for all the microgrid components. Each agent then exchanges power accordingly with the grid, and the storage system draws/injects a certain amount of power in the microgrid, hence getting to a new state at the subsequent time step. In the following paragraphs, we first overview the relevant work in this field and then list our contributions.

A. Related work

Learning-based MPC. The integration of MPC with ML techniques has recently gained significant attention in the control community for various purposes: notably, to enhance prediction models [41], to derive explicit MPC laws [34, 45], for controller auto-tuning [40, 27] and to forecast future profiles [46, 52]. In particular, for the EMS design problem, the synergistic application of MPC and learning methodologies has proven highly advantageous [61]. While MPC effectively compensates for uncertainties and handles constraints, learning tools are powerful for predicting uncertain profiles, such as load demand or renewable power generation. For example, in [46], Support Vector Machine regression is effectively employed to predict future renewable power generation and load demand. Authors of [26] incorporate a seasonal auto-regressive moving average model in the EMS to generate a set of future scenarios for a scenario-based MPC method.

Learning methods for microgrid EMS. NNs have emerged as valuable tools in this context because they can approximate functions with arbitrary accuracy [38]. More specifically, several examples exist in the literature where NNs are first offline trained based on available datasets and then used online for profile prediction [42, 46, 52, 58]. However, the main drawback of this approach is that these prediction tools are not able to adapt in real-time to possible changes in the environment, which can lead to issues in applications and scenarios involving ageing, sensor calibration, or the installation of new equipment [22]. Furthermore, specific plant datasets might not always be readily available beforehand. This paper adopts an OL approach to overcome these limitations, enabling the learning process to adapt in real-time. Though beneficial, this approach comes with some theoretical and implementation challenges. In [56], an ensemble combining offline and online learning is used for load forecasting. In [22], an OL Long Short-Term Memory (LSTM) model is presented, incorporating hyperparameter tuning to forecast load prediction. Online uncertainty estimation is included in [4] together with load forecasting. While achieving good results in online profile prediction, the tools above focus on the prediction of a single profile only, e.g. the load demand, and do not consider the EMS control problem. Moreover, one of the primary challenges of OL involves the trade-off between adaptation to new incoming information and retaining old knowledge, commonly referred to as catastrophic forgetting [25]. To deal with this issue, authors of [12] propose a renewable energy generation predictor that is retrained at the end of each day using all available data, resulting in a high computational cost. This paper proposes a method that allows us to adapt to new data while maintaining the memory of learnt information without requiring a high computational cost. In this context, Reinforcement learning (RL) has emerged as a promising approach for real-time controller adaptation under uncertainties. Its application in the EMS design has been explored in several papers, such as [31, 54, 37], which present methods for battery scheduling under uncertain conditions of load demand, renewable generation, and electricity price. Furthermore, in [49], an RL-based controller is proposed for EMS, demonstrating improved performance with respect to MPC in terms of real-time computation but at the cost of computationally expensive offline learning. However, it is essential to note that a crucial challenge in implementing

RL in real-world systems is that the RL exploration phase may lead to unsafe system states, requiring reliable simulation environments with ample available data, which is not always feasible or accurate. To address this issue, authors of [57] propose a safe exploration method that involves the implementation of an MPC-based safety filter after the RL controller is entitled to modify the input to maintain safe operation. Nevertheless, this approach requires adding an element to the controller, which is unsuitable for scheduling problems.In this respect, the proposed method allows us to take the best of the two worlds, the OL adaptation capabilities and the MPC constraints satisfaction ability, by integrating the control task in the learning framework. To obtain this, we consider a performance-based approach.

Performance-based Learning. The concept of performance-based learning, or E2E learning [33, 39], is closely intertwined with RL and focuses on generating prediction models with the objective of minimising decision errors rather than prediction errors when learning is used as an aid in decision making [17]. This approach has found application in scheduling tasks, like battery scheduling, as demonstrated in [15], where it shows superior performance in minimising a task-specific loss with respect to a controller trained to minimise a standard prediction error. However, in that work, the controller is not adapted online. Performance-based learning has also been considered for control purposes in [7], where a Bayesian optimisation [51] is used to select the best linear model for approximating the nonlinear dynamics of a robotic system, thereby maximising controller performance. Differentiable optimisation layers [2] have proven valuable in this context, enabling the learning of convex optimisation models [1] and policies [3]. An online implementation of this approach is proposed in [6], where a self-tuning MPC is developed using differentiable optimisation layers, as proposed in this paper. However, the model there is trained assuming the presence of an expert system (whose availability is often impossible to assume) running alongside it and minimising an imitation loss. Moreover, authors of [16] employ differentiable layers to obtain a deep learning alternative to explicit MPC. However, the training phase is performed fully offline, assuming the availability of a dataset of the system dynamics.

B. Contributions

This paper proposes a novel learning-based MPC control architecture for EMS that can adapt to changes in the environment thanks to a specifically designed online training methodology. As opposed to standard methods such as [46], where the predictor is offline trained on an a priori available dataset, here we present an OL algorithm to adapt the NN predictor to possible changes based on the stream of incoming data. The predicted profiles are then used to feed an optimisation layer that computes the power schedule over the prediction horizon in a receding horizon fashion.

To avoid catastrophic forgetting, without the need of retraining the network as in [12], we design novel O-SWA and ER algorithms, able to be online implemented for the first time in the literature, to the best of the Authors' knowledge. While the O-SWA implicitly maintains the knowledge gained in the previous steps by averaging new and past NN weights, the ER explicitly presents past input-output pairs to the network. By using differentiable optimisation layers [2], it is possible to differentiate through the constrained optimisation problem at the last NN layer to train the network in an E2E fashion, i.e. by tuning the weights of the NN targeting the closed-loop controller performance instead of the unknown profiles prediction or system identification loss. In this way, moreover, it is possible to enforce the satisfaction of some operation and safety constraints, embedding them in the NN. This paper represents the first time that differentiable layers are used online for an E2E implementation of a learning-based MPC without requiring an expert controller, in general, not only in the microgrids application. This is in contrast to [15, 16], where the controller is offline trained, and to [6], requiring an expert controller system running alongside it.

Hyperparameters are selected via the HPO algorithm suitable for online learning, employing a custom validation loss that can target the unknown profile prediction or the EMS performance. Moreover, since data of a specific microgrid may not be available a priori in the EMS design phase, and motivated by the fact that an online trained algorithm might have poor performance for some initial steps, we show that the proposed methodology has good TL [60] capabilities, allowing to pre-train the algorithm on any available microgrid dataset and obtain, from the beginning, better performance when online implementing the algorithm on the actual system. This shows that the method is suitable for implementation on real systems and provides additional evidence of the effectiveness of the OL algorithm in adapting from one microgrid to another. We then provide an estimate of the memory required for hardware implementation to show the feasibility of implementation in real-world scenarios. These measures are also used as performance indicators to compare controllers characterised by different choices of hyperparameters affecting the network size. Finally, we show that our method can easily be extended to online uncertainty estimation. In particular, we modify the regression loss function to estimate the quantiles of the unknown profiles. We use this additional information to estimate the cost uncertainty over the prediction horizon online to support the decision process.

Preliminary results have been presented in [8, 9, 10]. In this paper, we integrate some previously and separately proposed results in a comprehensive architecture and significantly extend the contributions. In particular, as opposed to [8, 9], in this paper, we propose a different task loss, not requiring the solution of one additional optimisation problem at each time step, thus reducing the computational effort. Moreover, we introduce a proper HPO procedure suitable for the online learning algorithm. We propose enhancements to the online learning algorithm, specifically the O-SWA and the ER procedures, and we explicitly consider the uncertainty in the system's parameters. Compared to [10], the main novelties lie in integrating regression and task losses within the online learning framework and the ER algorithm. Furthermore, we provide extensive simulation results. Finally, as a novel contribution with respect to all our previous papers, in this paper,

we introduce a method for online uncertainty quantification and consider the memory requirements for controller implementation.

Summarising, the two main contributions of the paper are: 1) a novel E2E task-based learning methodology that can be implemented online without requiring the availability of a priori training datasets or the presence of expert systems; 2) a novel microgrid EMS which estimates uncertain profiles of prices, demand and generation by jointly optimising prediction error and controller performance and can adapt to uncertainty and changes in the systems parameters and the environment.

Further novel contributions of this paper are:

- we propose an online implementation of the ER algorithm;
- we define a task-based loss for the online NN training;
- we explore the algorithm capabilities in the case of an uncertain system model, in particular, when the task-based loss is used;
- we formally optimise the hyperparameters of the NN targeting the closed loop controller performance and the profile prediction accuracy;
- we investigate the transfer learning capabilities when the task-based loss is used;
- we estimate an indicator for the memory required by different controller architectures;
- we extend the method for online uncertainty estimation, showing a possible application in the online estimation of the future cost.

The code used for the numerical validation in the simulation is available at `https://github.com/vittpi/ol-ems`.

The remainder of the paper is organised as follows. In Section II, we describe the considered model of the microgrid and the proposed EMS architecture. In Section III, we present the online training algorithm, the NN architecture, and the HPO procedure. In Section IV, we present the simulation results. Conclusions are drawn in Section V.

C. Notation

We use subscripts to denote time instants. For example, the variable v_t is the value of v at time step t. To denote the value of a variable k steps ahead of time t we use the short notation $v_{k|t}$, i.e. v_{k+t}. We use the hat to represent predicted values, e.g. $\hat{v}$ estimates the variable v. Time sequences are denoted using bold variables. Subscripts are used to state the sequence length. Hence the sequence of N samples of the variable v from time t is $\mathbf{v}_{N|t} = \{v_{0|t}, v_{1|t}, \cdots v_{N-1|t}\}$.

II. Problem formulation

In this Section, we describe the microgrid model and the proposed EMS architecture, as illustrated in Fig. 2.

A. Microgrid model

We consider a microgrid model composed of four types of agents, represented in the microgrid in Fig. 1: (i) renewable generators; (ii) loads; (iii) storage systems; (iv) utility grid connections. The work can easily be extended to other types of agents [11]. Renewable generators are the power sources in the grid; they collectively produce a power $P_t^r \geq 0$ at each time step. On the other hand, loads are the power sinks, characterised by their cumulative power demand $P_t^l \geq 0$. We assume loads and renewable generators are non-controllable, i.e., changing their power profiles is impossible. The controllable loads and renewable generators case is covered in [11]. There are many reasons to install storage systems in microgrids, from compensation for intermittence of renewable resources to enhancing power quality [20, 24]. Here, we are mainly interested in the peak shaving and valley-filling capabilities of storage systems for microgrid energy management. Hence, we model them as first-order linear systems, as commonly done in the EMS literature [46, 48], as follows:

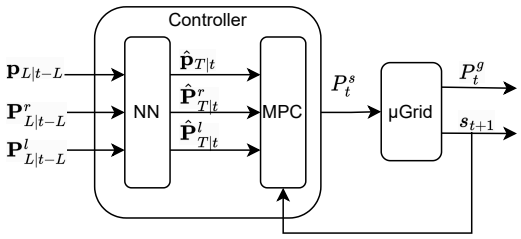

Fig. 2. The architecture of the EMS controller and the microgrid system.

$$s_{t+1} = (1-\sigma)s_t + T_s P_t^s \tag{1}$$

where $\sigma \in [0,1]$ is the self-discharge rate of the battery, T_s is the sample time of the controller, the power P_t^s is the sum of the charging and discharging power:

$$P_t^s = \eta_{in} P_t^{s_{in}} - \eta_{out} P_t^{s_{out}} \tag{2}$$

where $P_t^{s_{in}} \geq 0$, $P_t^{s_{out}} \geq 0$ are the input/output powers, and $\eta_{in} \in [0,1]\,, \eta_{out} \geq 1$ are the charging/discharging efficiencies respectively. The storage power and charge are limited as:

$$-P_M^s \leq P_t^s \leq P_M^s \tag{3}$$

$$s_m \leq s_t \leq s_M \tag{4}$$

where P_M^s is the maximum power that can be exchanged with the microgrid, s_M is the maximum charge level of the battery, and s_m is the minimum charge level of the battery. In this work, we assume all the agents are connected to a common bus connected to the utility grid. Hence, all the agents are coupled via the power balance constraint:

$$P_t^s + P_t^g = P_t^r - P_t^l \tag{5}$$

where P_t^g is the power exchanged with the utility grid ($P_t^g \geq 0$ when the microgrid is selling power).

B. EMS architecture

The architecture of the controller is represented on the left of Fig. 2. The controller takes as input the past values of the profiles of price renewable generation and loads over a look-back window of length L and uses them to predict their future values over a prediction horizon of length T. Other relevant

data can easily be included in the proposed architecture as NN inputs, if available, to obtain more reliable predictions; e.g. in [15], information about the next day's temperature forecast and binary indicators of weekends are used. The selection of features for prediction is out of the scope of this paper. The MPC block uses the predicted profiles to compute the power reference profile of the storage system. Subsequently, the microgrid system schedules the operation of all the agents connected to it by implementing the control action computed by the controller. Namely, once the optimal input $P_t^{s,\star}$ is computed by the controller, the system evolves according to Eqs. (6) and (7):

$$s_{t+1} = (1-\sigma)s_t + T_s P_t^{s,\star} \tag{6}$$

$$P_t^g = P_t^r - P_t^l - P_t^{s,\star}. \tag{7}$$

The next state of the storage is measured and fed back to the controller, whereas the power exchanged with the utility grid determines the cost that the microgrid pays. At each time step, the cost paid to the electricity grid is computed as:

$$\mathrm{C}_t = -p_t P_t^g, \tag{8}$$

where p_t is the energy price at time t and the minus sign is used since P_t^g is positive when power is sold to the grid. The goal of the energy management system is to minimise the total energy cost due to energy trading, that is:

$$\mathrm{C} = \sum_{t=0}^{\infty} \mathrm{C}_t \tag{9}$$

and its units are a currency (e.g. €).

III. Control architecture and methodology

In this Section, we introduce the novel OL-based MPC methodology for EMS, illustrated on the left-hand side of Fig. 2. The controller is an NN whose last layer is a differentiable optimisation layer [2] implementing the MPC optimisation. In the following subsections, we describe the proposed control architecture and methodology and provide details of the algorithms we developed to improve the controller's performance, allowing for real-time implementation.

A. Background on performance-based learning

In this paper, the so-called performance-based learning approach, successfully adopted in the ML literature [33, 39, 17], is exploited for control purposes by extending it for real-time implementation. In the performance-based framework, ML methods are used to improve the solution of partially defined optimisation problems:

$$\min_{\xi} \quad J(\xi, \omega) \tag{10a}$$

$$\textit{s.t.} \quad \xi \in \Gamma(\omega), \tag{10b}$$

where J is the objective function, ξ is the decision variable, Γ is a constraint set and ω is an unknown parameter vector. Assuming we have a dataset of N observations of input-output ψ_i-ω_i pairs $\mathcal{D} = \{\psi_i, \omega_i\}_{i=1}^N$, the goal is to use supervised learning to compute the best optimal solution $\xi^*(\hat{\omega})$, where $\hat{\omega}$ is the estimation of ω.

The standard approach, e.g. the one used in [46] in the EMS case, consists in first estimating the unknown parameter ω, for example, by training a NN based on the dataset $\mathcal{D}$ minimising the mean squared error loss between ω and $\hat{\omega}$, and secondly in using the estimate for the subsequent optimisation. In contrast, the optimisation objective is explicitly considered when defining the loss function used for NN training in the performance-based approach. We refer to this loss function as *task loss*, which can be defined in different ways, e.g. as the *regret* in [9]. This paper uses the optimisation objective function J of Eq. (10a) as task loss. There are two main difficulties in the implementation of performance-based learning for control. The first one is the differentiation through the *argmin* function required to compute the gradients of the loss $\mathcal{L}$ with respect to the NN weights θ for the backpropagation algorithm, i.e. computing $\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \xi^\star}\frac{\partial \xi^\star}{\partial \hat{\omega}}\frac{\partial \hat{\omega}}{\partial \theta}$. This has been done for different types of optimisation problems: quadratic programming [5], linear programming [17], combinatorial problems [23] and a Python toolbox is presented in [2]. In particular, in [5] an efficient method for convex problems has been proposed by differentiating the KKT conditions of the problem at its solution and this is the approach followed in this paper. The main limitation of this method is that the optimisation problem must be convex; however suitable convexification methods might be used if non-convex constraints need to be included. The second one is the online implementation for MPC. Since the controller has to work in real-time, the whole dataset $\mathcal{D}$ is not available beforehand, and the control action ξ has to be computed at each time step. Hence, an online performance-based learning algorithm is designed in this paper. Throughout the paper, we denote performance-based learning as E2E learning or task-based learning.

B. NN controller architecture

The first layers of the proposed NN are used to compute the prediction of the unknown price, load and renewable generator profiles, which we denote as $\hat{\mathbf{p}}_{T|t}$, $\hat{\mathbf{P}}^l_{T|t}$ and $\hat{\mathbf{P}}^r_{T|t}$ respectively, as in Fig. 2. The predictions are computed based on past values of the profiles in a look-back window of length L. Specifically, an LSTM RNN, firstly proposed in [29], is used due to its ability to capture dependencies in time series while maintaining a small number of parameters and flexible input time window. The main hyperparameters to be selected are the number of LSTM layers and the number of hidden units, which we denote as n_{layers} and n_h, respectively. This paper uses a single RNN to jointly predict all three unknown profiles to learn potential correlations among the input signals and obtain more accurate predictions. It is important to note that other NN architectures can be used in the proposed framework. For example, a different network could be used for each profile, as in [8], as well as other input data as in [15], or the prediction of one profile could be the input for the prediction of another in a cascaded fashion. However, since our goal is to develop a general framework for jointly optimising profile prediction and control performance, we did not introduce additional

biases in the architecture, and we kept the NN as simple as possible. At each time step the NN is fed with $\mathbf{p}_{L|t-L}$, $\mathbf{P}^l_{L|t-L}$ and $\mathbf{P}^r_{L|t-L}$, and outputs the predictions $\hat{\mathbf{p}}_{T|t}$, $\hat{\mathbf{P}}^l_{T|t}$ and $\hat{\mathbf{P}}^r_{T|t}$ over the MPC prediction horizon. The predictions are obtained by collecting the last hidden state of the RNN and feeding it to a dense layer. The NN output dimension is $T \times 3$, containing the predictions of the profiles over the MPC prediction horizon. In contrast, the input dimension is $L \times 3$, the past values of the profiles over the look-back window. Once the prediction is computed, it is passed to the subsequent constrained optimisation layer of the NN. The optimisation problem is reformulated at each time step considering the newly available state measurements and profile predictions. The optimisation problem can be written as:

$$\min_{\mathbf{P}_t^{s_{in}}, \mathbf{P}_t^{s_{out}}} \quad \sum_{k=0}^{T-1} -\hat{p}_{k|t} P^g_{k|t} \tag{11a}$$

$$\text{s.t.} \quad s_{k+1|t} = (1-\hat{\sigma}) s_{k|t} + T_s P^s_{k|t} \tag{11b}$$

$$P^s_{k|t} = \hat{\eta}_{in} P^{s_{in}}_{k|t} - \hat{\eta}_{out} P^{s_{out}}_{k|t} \tag{11c}$$

$$-P^s_M \leq P^s_{k|t} \leq P^s_M \tag{11d}$$

$$s_m \leq s_{k|t} \leq s_M \tag{11e}$$

$$P^g_{k|t} + P^s_{k|t} = \hat{P}^r_{k|t} - \hat{P}^l_{k|t} \tag{11f}$$

$$s_{0|t} = s_t \tag{11g}$$

In the above optimisation problem, some values are not known; hence, their estimate is used. In particular, the profiles over the prediction horizon of the electricity price $\hat{\mathbf{p}}_t$, renewable generator $\hat{\mathbf{P}}^r_t$ and load $\hat{\mathbf{P}}^r_t$ are estimated using the previous layers of the NN. Moreover, the parameters of the storage model are also supposed to be uncertain; hence, their estimate is used: $\hat{\sigma}$, $\hat{\eta}_{in}$ and $\hat{\eta}_{out}$. In this work, we focus on the estimation of the unknown profiles. At the same time, the uncertainty with respect to the storage parameters is addressed using the task-based approach as discussed in section III-C. The objective function (11a) is the finite horizon approximation of (9) over the prediction horizon computed using the predicted price profile. Constraints (11b)-(11f) represent the model of the microgrid and its agents, described in Section II-A. The last constraint (11g) is the state feedback. The decision variables of the optimisation problem are the input and output power exchanged with the storage, $\mathbf{P}_t^{s_{in}}, \mathbf{P}_t^{s_{out}}$. Once the optimal solution of Problem (11a)-(11g) is computed, the control law is defined as follows according to the MPC algorithm:

$$P_t^{s_{in}} = P_{0|t}^{s_{in}\star} \qquad P_t^{s_{out}} = P_{0|t}^{s_{out}\star} \tag{12}$$

and $P_t^{s,\star}$ is computed for Eq. (6) through Eq. (2).

C. Online learning

In this section, we describe the online training algorithm used to train the NN at each time step t, i.e. the procedure employed to update the weights of the NN from θ_{t-1} to θ_t. In particular, we describe the training and test sets and the loss functions employed at each time step. We denote the input tensor used at time step t as $\psi_t^* \in \mathbb{R}^{B^* \times L \times F_{\text{in}}}$, where $*$ is used to denote the training and test sets respectively, with $* \in \{tr, test\}$, B^* is the size of the batch of data used for $*$ and F_{in} is the number of input features. In this work, by *feature*, we mean the profile of price, load or renewable generation. Thus, we have the number of features $F_{\text{in}} = 3$ for our EMS application. The output tensor is denoted as $\omega_t^* \in \mathbb{R}^{B^* \times T \times F_{\text{out}}}$, where F_{out} is the number of output features. Therefore, $F_{\text{out}} = 3$ in the considered scenario. Fig. 3 illustrates the data splitting in an example where the look-back window size is $L = 4$, the MPC horizon window length is $T = 3$, and the batch size is $B_{tr} = 2$. We first present the training and test sets, then the loss function employed for NN training. At time step t, we first perform a training phase, i.e. the update of the NN weights from θ_{t-1} to θ_t. For the training phase, we use input and output samples of the unknown profiles that are fully in the past. Hence, the training set is as follows:

$$\psi_t^{tr} = \begin{bmatrix} \mathbf{v}_{L|t-T-L} \\ \vdots \\ \mathbf{v}_{L|t-T-L-j} \\ \vdots \\ \mathbf{v}_{L|t-T-L-B^{tr}+1} \end{bmatrix} \quad \omega_t^{tr} = \begin{bmatrix} \mathbf{v}_{T|t-T} \\ \vdots \\ \mathbf{v}_{T|t-T-j} \\ \vdots \\ \mathbf{v}_{T|t-T-B^{tr}+1} \end{bmatrix} \tag{13}$$

where $j = 0, \ldots, B^{tr} - 1$. After the training phase, the NN is used to predict the future values of the uncertain profile $\mathbf{v}_\bullet$ over the MPC prediction horizon to use them in the MPC optimisation. The test set input is composed of the most recent L samples, whereas the test set output is composed of the future T samples:

$$\psi_t^{test} = \begin{bmatrix} \mathbf{v}_{L|t-L} \end{bmatrix}, \quad \omega_t^{test} = \begin{bmatrix} \mathbf{v}_{T|t} \end{bmatrix}. \tag{14}$$

It is important to note that the test set outputs ω_t^{test} (which consist of the true future samples) are not used for the learning process at time t, nor for the control optimisation, as they are in the future and therefore unknown at time t: they are only employed a posteriori in simulation to assess the prediction performance and compare different architectures. The outputs of the NN at time t injected by the current input test set are the uncertain predictions for the future T samples and are the ones used in the MPC optimisation problem in the horizon window. At time step $t+1$, the test and training sets are shifted forward by one, and the process is repeated.

We now introduce the loss function employed for the NN training. We use as a loss function a weighted sum of regression and a task loss:

$$\mathcal{L}_t(\theta_{t-1}, \psi_t^{tr}, \omega_t^{tr}) = \sum_{v \in \{p, P^r, P^l\}} \alpha_v \mathcal{L}_t^{reg,v}(\theta_{t-1}, \psi_t^{tr}, \omega_t^{tr}) + \beta \mathcal{L}_t^{task}(\theta_{t-1}, \psi_t^{tr}, \omega_t^{tr}), \tag{15}$$

where $\mathcal{L}_t^{reg,v}$ is the regression loss for feature v, $\mathcal{L}_t^{task}$ is the task loss, $\alpha_v \in \mathbb{R}$ is a weight associated to each feature and $\beta \in \mathbb{R}$ is the weight associated to the task loss. These parameters can be optimised during the HPO presented in

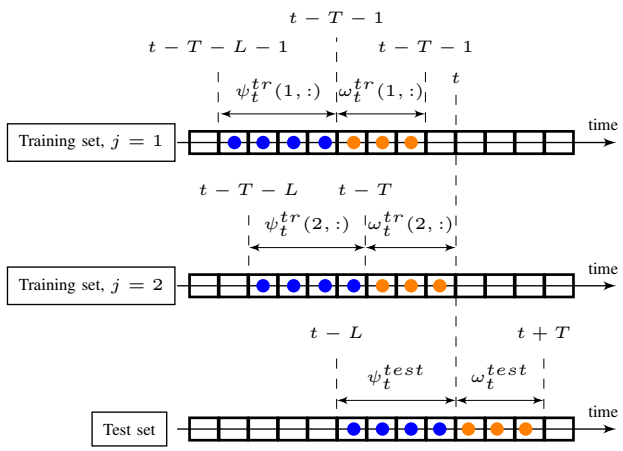

Fig. 3. Representation of the training and test sets for $L = 4$, $T = 3$, $B_{tr} = 2$ at time t. The top and middle rows represent the first and second training batches. The bottom row represents the test set at time t. Each black square represents a feature sample f_t. Samples highlighted in blue are the inputs of the NN, and the ones in orange are the NN's desired outputs.

Section III-F. Firstly, the regression loss at time t is defined as the MSE loss for each feature:

$$\mathcal{L}_t^{reg,v}(\theta_{t-1}, \psi_t^{tr}, \omega_t^{tr}) = \sum_{j=1}^{B^{tr}} \sum_{k=0}^{T-1} \left(v_{k|\bar{t}_j} - \hat{v}_{k|\bar{t}_j}\right)^2, \quad (16)$$

where $\bar{t}_j = t - T - j$ is introduced for the sake of clarity. Secondly, we define the task loss as:

$$\mathcal{L}_t^{task}(\theta_{t-1}, \psi_t^{tr}, \omega_t^{tr}) = \sum_{j=1}^{B^{tr}} \sum_{k=0}^{T-1} p_{k|\bar{t}_j} \hat{P}^g_{k|\bar{t}_j} \quad (17)$$

where the predicted profile of power exchanged with the utility grid $\hat{\mathbf{P}}^g_{T|\bar{t}_j}$ is computed as:

$$\hat{\mathbf{P}}^g_{T|\bar{t}_j} = \hat{\mathbf{P}}^r_{T|\bar{t}_j} - \hat{\mathbf{P}}^l_{T|\bar{t}_j} - \mathbf{P}^{s,\star}_{T|\bar{t}_j}. \quad (18)$$

In other words, by minimising the regression loss (16), we aim at improving the prediction performance for the price, renewable generation, and load profiles as new data is collected. Improved predictions allow one to compute the control action in (11a)-(11g), which minimises the total electricity cost while guaranteeing the defined operation and safety constraints. On the other hand, the task loss (17) targets the electricity cost directly. The reason behind the choice of integrating the task loss, and therefore to implement an E2E or task-based approach, is to allow the learning process to implicitly take into account other potential sources of uncertainties in the parameters of the system or external disturbances, such as the energy conversion efficiencies for example, which would not be estimated by a regression-loss-only trained NN but affect the EMS performance. Computing the task loss downstream of the optimisation problem implicitly allows us to consider these unknown parameters. The advantages of employing the task loss in the online training are shown in simulation results Section IV-C, where more details are provided.

Remark 1: It is important to note that the task loss is defined as the electricity cost over the prediction horizon under the true price profile $p_{k|\bar{t}_j}$. The electricity price profile is fully available at time t since we only perform the training phase using past samples. The subscript of the price in (17) is $k|\bar{t}_j$, which denotes only samples in the past with respect to t.

D. Online stochastic weight averaging

The main objective of the OL approach is to adapt to rapid environmental variations, such as fluctuations in electricity prices, while preserving the knowledge gathered in previous steps. To bolster the learning process' resilience to abrupt environmental changes and input signals, we adapt the Stochastic Weight Averaging (SWA) method [30] for implementation in OL-based controllers. SWA is commonly used to enhance the generalisability of NNs in an offline learning context. It defines the final model as the average across the last N training iterations. However, in this work, we modify the algorithm to fit an OL context, where an average model is updated at every instance and denoted as O-SWA.

We symbolise θ_t^{∇} as the weights post-gradient update at instance t. γ serves as the moving average decay factor, while $\theta_t^{avg}, \theta_{t+1}^{avg}$ represent the moving averages of the weights at instances $t, t+1$, respectively. The weights for the subsequent instance θ_{t+1} are updated every δ steps. Initially, we set $\theta_0^{avg} = \theta_0$, and subsequently, at each instance, we revise both the average and current models via a moving average. In the OL deployment context, updating the average weights θ_t^{avg} in t is required at every instance since we cannot anticipate the last N training iterations as in [30]. Hyperparameters γ and δ can be adjusted to modulate the learning algorithm's sensitivity. These parameters govern the degree to which the current model relies on the average model θ_t^{avg}, a smoothed variant of the current model θ_t. The parameter γ regulates the proportion of preserved weights. For instance, $\gamma = 0.9$ indicates that 0.9 of the average's magnitude is maintained, and 0.1 is the weighting of the current model at time step t. The parameter δ controls the frequency with which the current model is replaced with the average model; for instance, $\delta = 10$ implies that every 10 time steps, we reset the current model to be equivalent to the average model. The purpose of setting the current model θ_{t+1} to the average model θ_{t+1}^{avg} is to encourage the NN's generalisability by resetting its optimisation pathway. This helps the network to avoid becoming entrapped in local minima or to be overly sensitive to sudden changes in the input signals.

E. Experience Replay

ER [13] is a method designed to make learning more robust and generalisable over time by revisiting experiences encountered at previous time steps. It operates by maintaining a buffer $\mathcal{B}$ of $|\mathcal{B}|$ past examples, which can be updated at each time step t with newly observed examples. In this work, we consider the buffer incrementally updated by reservoir sampling [55]. Reservoir sampling is a randomised algorithm for selecting k from an unknown population of N samples. This makes it ideal for the OL scenario since the number of time steps is unknown.

Given the currently observed data ψ_t^{tr}, and data sampled from the buffer $\psi_t^{\mathcal{B}}$, we update the model with respect to a

loss function and current model $\mathcal{M}(\theta_t)$ and both the current and past data $\psi_t^{tr} \cup \psi_t^{\mathcal{B}}$. The ultimate value of the loss function at each time step t is a linear combination of the loss over the current and past data, weighted through the hyperparameter ϵ. After the training step, reservoir sampling adds the current data to $\mathcal{B}$.

While adept at smoothing over rapid changes and avoiding local optima, the O-SWA is fundamentally weighting the previous experiences less over time. Integrating ER with O-SWA creates a complementary learning system that is resilient to rapid fluctuations and cognisant of an extensive range of past experiences. It strengthens the robustness and adaptability of our learning system, thereby ensuring optimal performance even in highly dynamic environments. On the one hand, ER collects a more comprehensive set of experiences to learn from, increasing sample efficiency and mitigating issues like catastrophic forgetting [25]. On the other hand, O-SWA improves generalisation by capturing all weight configurations throughout training rather than relying on the final weight set that could be specific to the most recent experiences. Combining these two approaches offers a method that efficiently uses training data and generalises well to new situations. The complete Algorithm 1 illustrates both methodologies together.

Algorithm 1 Online Training with O-SWA and ER

Input: $\theta_0, \gamma, \delta, \theta_0^{avg} = \theta_0, \alpha, \mathcal{B} = \emptyset, t = 0$
for t = 1,2,... **do**
  Receive current data ψ_t^{tr}
  Sample past data $\psi_t^{\mathcal{B}}$ from $\mathcal{B}$
  Compute loss $\mathcal{L}_t(\theta_{t-1}, \psi_t^{tr}, \omega_t^{tr}) + \epsilon \mathcal{L}_t(\theta_{t-1}, \psi_t^{\mathcal{B}}, \omega_t^{\mathcal{B}})$
  Perform gradient update and obtain θ_t^{∇}
  $\theta_t^{avg} = \gamma \theta_{t-1}^{avg} + (1-\gamma)\theta_t^{\nabla}$
  if $t+1$ mod $\delta = 0$ **then**
    $\theta_t = \theta_t^{avg}$
  else
    $\theta_t = \theta_t^{\nabla}$
  end if
  Add ψ_t^{tr} to $\mathcal{B}$ using reservoir sampling
  Compute and implement the control action
end for

At time step $t = 0$ the Algorithm 1 first initiates weights θ_0, decay factor γ, the time interval for weight update δ, and an empty buffer $\mathcal{B}$ for past experiences. At t it extracts additional training data from the past samples that have been stored in the experience buffer in the previous time steps $\psi_t^{\mathcal{B}} \sim \mathcal{B}$ and uses it to replay past experiences to the NN and improve its generalisation capabilities. If the buffer is not populated, no data is fetched from the buffer, and the related operations to $\psi_t^{\mathcal{B}}$ are skipped. The algorithm then computes a combined loss, a weighted average of losses from the current and past data, governed by a hyperparameter α. Subsequently, the gradient update is performed using this calculated loss. The next step involves updating the average model weights using a decay factor γ. Depending on the predefined time interval δ, the model decides whether to use the average weights θ_{t+1}^{avg} or the gradient-updated weights θ_t^{∇} for the next time step $t+1$. After this, the current data is added to the buffer using a reservoir sampling technique, and the time step is incremented. This approach amalgamates the benefits of O-SWA and ER, offering an efficient learning method that adapts quickly to environmental changes and prevents catastrophic forgetting by effectively leveraging past experiences.

F. Hyperparameter optimisation

To enhance the efficacy of the learning, it is crucial to tune the hyperparameters of the algorithm and identify the optimal NN architecture for the specific considered problem as well as the hyperparameters $\gamma, \alpha_v, \beta, |\mathcal{B}|, \delta$, ϵ, n_h, learning rate and weight decay. Moreover, the HPO algorithm has to be suitable for the proposed online learning algorithm presented in Section III-C, i.e. the winning hyperparameters have to work optimally when implemented online. We frame the HPO task in terms of a domain dataset, denoted as $\mathcal{D}_1$, a predefined wall-clock time budget C, for example the HPO will run for a maximum of 3 hours, and a validation loss $\mathcal{L}_t^{val}$. We assume the set $\mathcal{D}_1$ is available before the deployment of the EMS, for example, recordings coming from another microgrid with similar characteristics. The a priori available dataset $\mathcal{D}_1$ is used to initialize the weights of the NN and to validate the hyperparameters. As for validation loss, we decided to consider either the regression loss $\mathcal{L}_t^{reg}$ or the task loss $\mathcal{L}_t^{task}$ respectively when we use the regression or the task loss for training, i.e.:

$$\mathcal{L}_t^{val} = \begin{cases} \sum_{v \in \{p, P^r, P^l\}} \mathcal{L}_t^{reg,v}(\theta_{t-1}, \psi_t^{val}, \omega_t^{val}) & \text{if } \beta = 0 \\ \mathcal{L}_t^{task}(\theta_{t-1}, \psi_t^{val}, \omega_t^{val}) & \text{if } \beta \neq 0 \end{cases} \tag{19}$$

The training and validation sets for the HPO are selected by employing sample sequences that are fully in the past. Given a training batch size B^{tr} and a validation batch size B^{val}, the HPO training and validation batches are: $\psi_t^{tr} = \left[\mathbf{v}_{L|t-T-L-B^{val}-j}\right]$, $\omega_t^{tr} = \left[\mathbf{v}_{T|t-T-B^{val}-j}\right]$ for $j = 0, \ldots, B^{tr}$ and $\psi_t^{tr} = \left[\mathbf{v}_{L|t-T-L-B^{val}-i}\right]$ $\omega_t^{tr} = \left[\mathbf{v}_{T|t-T-B^{val}-i}\right]$ for $i = 0, \ldots, B^{val}$. Our goal is to find the optimal hyperparameters and architecture that yield the lowest value of a chosen loss function $\mathcal{L}_t^{val}$, within the time budget C using the validation set from $\mathcal{D}_1$. This optimisation process results in a winning model $\mathcal{M}^\star$ and its associated hyperparameters. Furthermore, the winning model $\mathcal{M}^\star$ and its configuration can serve as an initialisation point for learning on a dataset $\mathcal{D}_2$, offering potential advantages for Transfer Learning (TL) scenarios. TL capabilities are explored in Section IV.

G. Estimating uncertainty

In this Section, we present a methodology to assess the confidence of the learning-based predictions and provide a real-time indicator of the reliability of the prediction of the NN controller. In general terms, an NN predictor, by using an MSE loss function for training, focuses on the mean of the prediction error, thus urging the network to learn the expected value of the output target variable. However, the

target distribution's variance, or other uncertainty measures, often hold important information. To estimate uncertainty, an alternative to what was previously introduced, we can train the NN predictor online using the quantile loss function defined in [32] as:

$$\mathcal{L}_t(\tau, \hat{y}, y) = \sum_{i=1}^{N} \rho_\tau(\hat{y}_i - y_i), \tag{20}$$

where $\hat{y}_i$ denotes the predicted value and y_i is the target value. The function ρ_τ, parameterised by the quantile τ, is defined:

$$\rho_\tau(x) = \begin{cases} \tau x & \text{if } x \geq 0 \\ (\tau - 1)x & \text{if } x < 0 \end{cases} \tag{21}$$

By minimising the quantile loss function during training, the NN learns the quantiles of the output distribution. Consequently, we expand the output layer of the NN predictor to $2 \times N_{\text{quantiles}}$ neurons, where $N_{\text{quantiles}}$ is the count of quantiles that need to be learned. In our default configuration, we set $N_{\text{quantiles}} = 3$, which corresponds to the 10^{th}, 50^{th}, and 90^{th} quantiles. Here, the 50^{th} quantile represents the median of the target distribution, while the 10^{th} and 90^{th} quantiles signify the lower and upper limits of a predetermined confidence interval, respectively. In our implementation, the NN predictor is trained to minimise the quantile loss function in (20) for each quantile, resulting in a loss that is an equally weighted accumulation of individual quantile losses. This approach promotes a comprehensive understanding of the underlying target distribution, thus allowing the enhancement of the reliability of the proposed OL-based MPC, providing a support decision tool able to evaluate the uncertainty in the prediction and real-time evaluation of the cost over the prediction horizon. In this work, in the Numerical Results Section IV-E, we use the uncertainty measure to estimate the cost over the prediction horizon. For example, load agents can use this estimation to adjust their power demand depending on the predicted cost.

H. Real-world deployment

While in this work, we do not consider any particular hardware restriction for controller implementation. We still aim to design an efficient controller architecture that could be deployed in real-world systems. Moreover, NNs implementation may require large memory units to store the weight of the network, possibly making their deployment uneconomical. Hence, the memory requirements have to be adhered to starting from the controller algorithm design. To guide our architecture choices, we provide estimates of the memory requirements for our controller's NN, considering the number of parameters needed for the prediction horizon, look-back window, and other hyperparameters. Though not strictly constrained, staying mindful of potential hardware limitations will allow us to balance controller performance with practical implementability.

In the following, we compute the total number of samples stored at each time step by the proposed OL NN controller, depending on the specific architecture choices. This will be used in the next Section to compare the performance of different designed controllers. Firstly, the number of input-output samples is computed as:

$$N_{i,o} = F_{in} \times L + F_{out} \times T. \tag{22}$$

In case the ER algorithm presented in Section III-E is employed with a buffer of size $|\mathcal{B}|$, then an additional number of input-output pairs needs to be stored:

$$N_{\mathcal{B}} = N_{i,o} \times |\mathcal{B}|. \tag{23}$$

Naming the number of NN weights that need to be stored as N_θ, the O-SWA requires the storage of additional N_θ weights. As shown in Section IV, the main parameters influencing the required parameters are the LSTM hidden dimension n_h and the buffer size $|\mathcal{B}|$. The previous equations allow us to consider the scalability of the method with respect to the number of parameters like the look-back window and MPC prediction horizon. If additional agents join the microgrid, the memory requirements can still be estimated through Eq. (22) by increasing the number of input/output features (one additional profile for each additional renewable generator or load agent). Moreover, the optimisation problem (11a)-(11g) has to be modified by introducing specific constraints for the additional agent (e.g. (3) and (4) for a storage system) and including the power profile of the agent in the power balance (11f).

Remark 2: We are proposing a centralised approach where it is possible to aggregate agents of the same type; in this sense, the proposed methodology can easily take an increasing number of agents into account, for example, by considering the sum of the loads as a single load. In this case, the computational complexity of the method would not change if more agents were considered.

To clarify the procedure for the controller deployment, Algorithm 2 lists all the required steps. Step 1 involves the optimisation of the hyperparameters and the pre-training on a dataset $\mathcal{D}_1$ (optional), including the uncertain profiles of price, demand and generation possibly available a priori (even from a different microgrid system), where $\tilde{\theta}$ represents the initial NN weights that are randomly initialised. Subsequently, the winning model $\mathcal{M}^\star$ with weights θ_0 is deployed in the actual microgrid represented by the dataset $\mathcal{D}_2$, where θ_0 are the NN weights after the pre-training phase. The following steps 2-7 represent the controller's online training and operations to compute the control action. Finally, the total cost can be computed to assess the controller's performance.

IV. NUMERICAL RESULTS

In this section, we show the performance of the proposed OL-based MPC for EMS and analyse the advantages of each proposed tool and methodology. All simulations were repeated with different random seeds, and the results were given in terms of mean and single standard deviation. We developed the simulations in Python 3.9.16 and the code, that will be available on GitHub at `https://github.com/vittpi/ol-ems`, uses PyTorch 1.10.1 [47], PyTorch Lightning 1.5.10 [21], syne-tune 0.2 [50] and cvxpylayers 0.1.5 [2]. The optimisation problem (11a)-(11g) is convex; hence, it is modelled using CVXPY [14] and solved using SCS

Algorithm 2 Controller implementation

Offline phase (optional): HPO and/or pre-training
Input: $\mathcal{D}_1$, $\mathcal{M}$, $\bar{\theta}$
HPO and pre-training on dataset $\mathcal{D}_1$

Online phase: training and input computation
Input: $\mathcal{D}_2$, $\mathcal{M}^\star$, θ_0
for t = 1,2,... **do**
  Online training step (Algorithm 1): $\theta_{t-1} \leftarrow \theta_t$
  Retrieve current state s_t
  Compute control action $\longrightarrow \mathbf{P}_t^{s_{in}}, \mathbf{P}_t^{s_{out}}$
  Apply control action $P_t^{s_{in}}$ $P_t^{s_{out}}$ $\implies$ $s_t \leftarrow s_{t+1}$
end for

Compute performance
Total cost (9)

solver [43]. The considered microgrid comprises the agents described in Section II-A. The comparison among different simulations is based on two performance indicators: (i) the total cost over the simulation steps computed by (9); and (ii) the MSE on the test set for all the involved features:

$$\mathrm{C}^{\mathrm{MSE}} = \sum_{v \in \{p_t, P_t^r, P_t^l\}} \sum_{t=0}^{S} \sum_{k=0}^{T-1} \left(v_{k|t} - \hat{v}_{k|t}\right)^2. \qquad (24)$$

In the following subsection, we present the datasets considered for the simulations. The time required for the execution of one step (online training and computation of the control action) is 0.07 seconds using an Intel® Core™ i7-2600 CPU @ 3.4GHz and 16GB RAM for all the simulations, making this method compliant with the EMS application where the sampling time of the controllers is in the order of minutes [59]. We then describe, in subsection IV-B, the results obtained by applying the OL algorithm presented in Section III-C, using only the regression loss (16). The goal is to show that the algorithm can learn online and adapt the controller to environmental changes. The advantages of O-SWA and ER algorithms presented in paragraphs III-D and III-E respectively, to improve the performance of the OL methodology are then shown in IV-B. In subsection IV-C, we present the effectiveness of using the task loss (17) for training the controller. We then explore the capabilities of the proposed TL algorithm in Section IV-D. Finally, in subsection IV-E, we show the results related to the estimation of quantiles of the electricity price.

A. Dataset description

We used the profiles of the electricity price, load demand and renewable generator for the simulation results from two different datasets. The first data source is the *EMSx* benchmark dataset collected by Schneider Electric, presented in [36], that collects the power profiles of loads and renewable generators and provides the parameters of the storage systems. Among the 70 industrial sites, we randomly picked site 10 and site 12, specifically the recordings between 01/01/2016 and 07/10/2017. The storage parameters $(s_M[\mathrm{kW\,h}], \eta_{in}[-], \eta_{out}[-], P_M^s[\mathrm{kW}])$ are $(400, 0.95, 1.05, 100)$ and $(800, 0.95, 1.05, 200)$ for site 10 and 12 respectively. The electricity price profiles instead are downloaded from the *ENTSO-E Transparency Platform* [19], described in [28]. The website collects the day-ahead electricity price for different European bidding zones. Specifically, we employed the profiles of *IT-Centre-North* of the same period of the power profiles. All the profiles are normalised in the 0-1 range; even though normalisation factors may not be available beforehand in the OL framework, they can be inferred from past recordings. Moreover, TL simulations show that good results can be achieved even if the normalisation factors do not squeeze all the samples in the specified range. The other numerical values used for simulations are: $T_s = 1\mathrm{h}$, $T = 24$, $L = 168$, $s_m = 0.1 s_M$, $\sigma = 0.0042$, $s_0 = 200\mathrm{kW\,h}$, $B^{tr} = B^{val} = 1$, $n_{\text{layers}} = 1$.

B. Online learning

This Section presents the simulation results related to OL, O-SWA and ER. The OL algorithm aims to adapt the prediction to changes in the profiles. Results are presented in Table I and Fig. 4. As opposed to [10], here the results are obtained considering uncertain storage model, i.e. setting $\hat{\sigma} = 0$ in (11b) and $\hat{\eta}_{in}, \hat{\eta}_{out} = 1$ in (11c). The hyperparameters for each industrial site are selected using the procedure explained in III-F using a random search algorithm with 4 CPU workers for a maximum time of 3 hours. The optimised hyperparameters are the learning rate, the weight decay, δ, γ and n_h and results are respectively $(3.38 \times 10^{-4}, 1.88 \times 10^{-4}, 400, 7.07 \times 10^{-1}, 48)$ and $(3.31 \times 10^{-3}, 8.88 \times 10^{-5}, 50, 8.00 \times 10^{-1}, 48)$ for site 10 and 12. In Fig. 4, the blue line represents the ground truth price profile. The orange line is the prediction obtained using our OL algorithm for the entire simulation length. The green line is the prediction obtained using our OL algorithm, but stopping the learning process at $\bar{t}$ after one year of training, on the 30th of December 2016, a day denoted by a red vertical line. In other words, the NN weights θ_t after $\bar{t}$ are $\theta_{\bar{t}+k} = \theta_{\bar{t}}$. After this date, when the price starts increasing, the figure shows that our OL algorithm can adapt the prediction, which takes approximately one day to adjust. The predictor that is not adapting online instead maintains the prediction on the same range as before $\bar{t}$, hence achieving a worse prediction performance. A similar simulation has been considered for the prediction of the load demand: Fig. 5 shows how the proposed methodology (orange) allows the load demand prediction to adapt when some new machinery is installed in the microgrid and the load changes subsequently. Instead, in the case that we do not use our proposed adaptive method and employ an NN predictor trained with the data till March 8 (red vertical line), the predictor does not adapt to the change.

This result is also evident from Table I. The first two lines of each simulation, OL and 1 year OL, report the total cost and the MSE of the two controllers for the two datasets, respectively. Using the OL algorithm allows the adaptation of the predictor to changes in the profiles, hence achieving a lower MSE and, therefore, a lower cost. We highlight that in this case, only the regression loss (16) is used; hence, the controller that predicts better the unknown profiles achieves the best performance in terms of electricity cost.

The third line for each site of Table I shows the results obtained disabling the O-SWA. To compare the results with the other cases fairly, we re-run the HPO, optimising all the other hyperparameters except the ones related to the SWA. Disabling the O-SWA increases the variability of the results and the mean value of the price with respect to the OL case.

The last line of Table I, denoted as "Prescient", shows the total cost obtained by an ideal controller when the true future values of the unknown profiles are used instead of the predicted one. In this case, the estimation error C^{MSE} is zero. The prescient controller's performance is the best. However, this scenario is ideal and cannot be realised in practice, but this controller represents the baseline against which to compare the novel proposed controllers given a prediction horizon T.

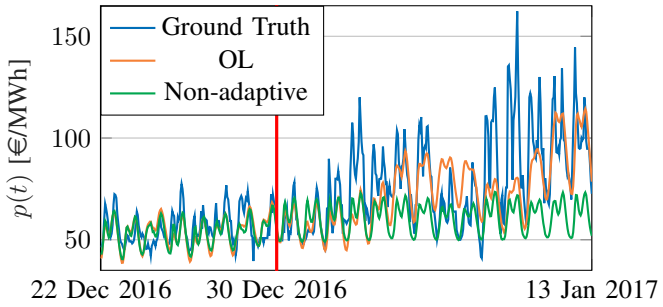

Fig. 4. Comparison of the performance of the proposed online learning-based predictor (OL) and a non-adaptive predictor where the training is stopped in correspondence with the red line (December 30). When the price starts increasing, our predictor (orange) adapts to the change.

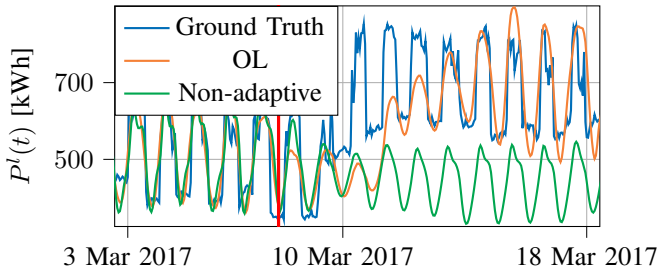

Fig. 5. Comparison of the performance of the proposed online learning-based predictor (OL) and a non-adaptive predictor where the training is stopped in correspondence with the red line (March 8). A new machinery is installed in the microgrid on March 10 causing a change in the load profile and a subsequent adaptation of the adaptive predictor.

We then tested the ER algorithm presented in Section III-E. We aim to show that using a buffer of $|\mathcal{B}|$. It is possible to obtain better results with respect to the ones presented in Table I for the OL case. Numerical results in Table II show that with a buffer of size $|\mathcal{B}| = 200$ and $|\mathcal{B}| = 100$ for site 10 and 12, respectively, it is possible to obtain better results with respect to the OL case of Table I. This good result comes at the cost of storing a number of samples that can be computed via (23) and computing the loss for the buffer samples.

TABLE I
OL RESULTS

Site	Controller	C [€]	C^{MSE}
10	OL	286,849 ± 149	1.234 ± 0.015
	1 year OL	288,072 ± 306	1.654 ± 0.018
	NO SWA	287,220 ± 325	1.323 ± 0.059
	Prescient	280,922	0.000
12	OL	61,578 ± 195	1.220 ± 0.005
	1 year OL	63,869 ± 385	1.681 ± 0.017
	NO SWA	62,480 ± 466	1.260 ± 0.029
	Prescient	49,781	0.000

TABLE II
EXPERIENCE REPLAY RESULTS

Site	B	C [€]	C^{MSE}
10	100	286,797 ± 192	1.103 ± 0.016
	200	286,563 ± 47	1.112 ± 0.009
	300	286,721 ± 108	1.095 ± 0.018
12	100	60,949 ± 45	1.095 ± 0.021
	200	61,438 ± 337	1.091 ± 0.015
	300	61,422 ± 502	1.102 ± 0.014

C. E2E learning simulations

In this Section, we show the advantages of using the task and the regression loss to train the NN. To do this, we ran simulations for both sites 10 and 12, enabling the task loss, i.e. setting $\beta \neq 0$ in (17), setting n_h to different values in the range $[2; 48]$. To fairly compare the results, HPO was done for all the NNs. The optimised hyperparameters are the learning rate, the weight decay, parameters α_v of (16) for each feature of the regression loss and β of the task loss (17). In the previous case, the O-SWA is active with hyperparameters δ and γ obtained for each site while the ER buffer is not used. For the controller trained with task loss only, we defined the HPO loss as the task loss on the validation dataset to find the best hyperparameters to minimise the total electricity cost. Fig. 6 shows the number of parameters N_θ required by the NN given the LSTM hidden dimension n_h. This information compares the controllers and estimates the memory requirement for hardware implementation for storing the NN parameters.

Tables III, IV, V and VI present the simulation results, comparing the performance in terms of total electricity cost, for the two industrial sites, in the case the model of the storage (i.e. the values of σ, η_{in} and η_{out}) is known –Tables III, V) or not –Tables IV, VI). The first column of the tables shows the number of hidden units of the LSTM layer employed for the simulation. The second column presents the results obtained by performing the training only minimising the regression loss (16) – i.e. setting $\alpha_v \neq 0$ and $\beta = 0$ in (15). The third column instead presents the results obtained when the sum of regression (16) and task loss (17) are jointly used for the training – i.e. both $\alpha_v, \beta \neq 0$ in (15).

We can see that task loss provides an advantage in all the cases considered by reducing total costs and variability. When the parameters of the storage model are not known, the following values are used: $(\sigma, \eta_{in}, \eta_{out}) = (0, 1, 1)$. For the

TABLE III
RESULTS OF SITE 10 WITH NO MODEL UNCERTAINTY

	Training loss	
n_h	regression only	regression + task
2	$288,124 \pm 50$	$288,060 \pm 190$
4	$287,587 \pm 42$	$287,783 \pm 146$
8	$288,576 \pm 250$	$286,994 \pm 307$
16	$286,961 \pm 136$	$286,382 \pm 54$
24	$286,781 \pm 330$	$286,632 \pm 51$
32	$286,552 \pm 257$	$286,075 \pm 131$
36	$287,349 \pm 304$	$286,215 \pm 148$
48	$285,874 \pm 76$	$286,209 \pm 229$

reader's convenience, Figures 7 and 8 from Tables III and VI, respectively, are provided. Fig. 7 shows the results of Table III and the obtained C^{MSE} for site 10 as a function of the NN hidden dimension when no model uncertainty is considered. Fig. 8 shows the results of Table VI and the obtained C^{MSE} for site 12 as a function of the NN hidden dimension when the parameters of the storage are not assumed to be known. Blue lines identify the results obtained using regression loss only. Orange lines identify results obtained using the task and regression loss.

The first outcome of the simulations can be deduced from Tables IV and VI: when the parameters of the storage are not known with certainty, by using both regression and task losses allows to achieve a lower electricity cost, especially with small n_h. Intuitively, this is because, being the loss calculated after the optimisation problem, which includes the storage system model, it is possible to learn the uncertain parameters implicitly. Secondly, from Tables III and V, we see that even when we assume to know perfectly the parameters of the storage system, the controller trained with task loss can achieve better performance for most n_h. For example, for $n_h = 48$, the controller trained with regression loss achieves a lower cost for both datasets. This result complies with the results presented in Section IV-B and [10], where the HPO procedure selects 48 as the optimal LSTM hidden dimension. The total cost using the task loss instead decreases until $n_h = 32$ and then increases again, possibly due to overfitting. This suggests that the optimal n_h when the task loss is used is lower than the optimal hidden dimension for profiles prediction. Hence, if limited hardware memory is available for controller implementation, training the controller with task loss is beneficial for better performance, especially in the relevant case when model parameters are uncertain. Finally, Fig. 7 shows the prediction accuracy and total cost obtained for site 10 when the storage model is assumed to be known. One interesting fact of these results is that, even though the storage parameters are known, a better performance in terms of cost can be obtained even with a higher C^{MSE}. E.g. for $n_h = 4$ and $n_h = 24$, the C^{MSE} is higher for the controller trained with task and regression loss. However, the total cost is lower, meaning that a good profile prediction accuracy does not necessarily lead to better ultimate performance, even if the system model is known.

TABLE IV
RESULTS OF SITE 10 WITH MODEL UNCERTAINTY

	Training loss	
n_h	regression only	regression + task
2	$291,444 \pm 972$	$289,792 \pm 1107$
4	$289,770 \pm 453$	$288,852 \pm 816$
8	$290,261 \pm 731$	$287,845 \pm 570$
16	$288,668 \pm 508$	$287,180 \pm 156$
24	$287,662 \pm 372$	$286,856 \pm 356$
32	$287,432 \pm 574$	$286,821 \pm 178$
36	$289,072 \pm 560$	$286,827 \pm 187$
48	$286,849 \pm 149$	$286,741 \pm 37$

TABLE V
RESULTS OF SITE 12 WITH NO MODEL UNCERTAINTY

	Training loss	
n_h	regression only	regression + task
2	$65,200 \pm 1,063$	$64,050 \pm 188$
4	$63,259 \pm 298$	$62,931 \pm 652$
8	$64,481 \pm 389$	$61,996 \pm 419$
16	$60,979 \pm 244$	$60,792 \pm 143$
24	$61,107 \pm 815$	$60,069 \pm 316$
32	$60,773 \pm 691$	$60,021 \pm 155$
36	$62,742 \pm 595$	$60,513 \pm 972$
48	$59,999 \pm 272$	$61,144 \pm 309$

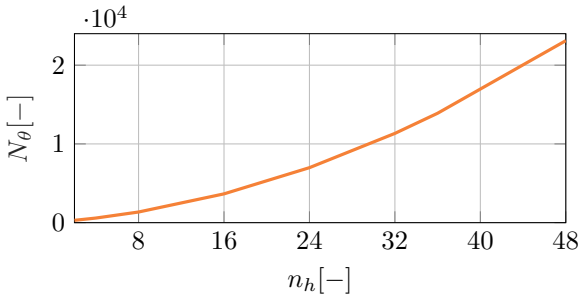

Fig. 6. Number of NN parameters N_θ as function of the LSTM hidden dimension n_h.

D. Transfer learning simulations

We now show the TL capabilities of the algorithm, which is particularly useful when a dataset of a specific microgrid is

TABLE VI
RESULTS OF SITE 12 WITH MODEL UNCERTAINTY

	Training loss	
n_h	regression only	regression + task
2	$68,841 \pm 1,465$	$68,953 \pm 2,134$
4	$66,006 \pm 283$	$65,474 \pm 360$
8	$67,097 \pm 742$	$63,470 \pm 979$
16	$63,097 \pm 308$	$63,074 \pm 212$
24	$63,595 \pm 357$	$61,924 \pm 329$
32	$62,225 \pm 241$	$61,603 \pm 191$
36	$65,889 \pm 489$	$61,571 \pm 246$
48	$61,578 \pm 195$	$62,531 \pm 199$

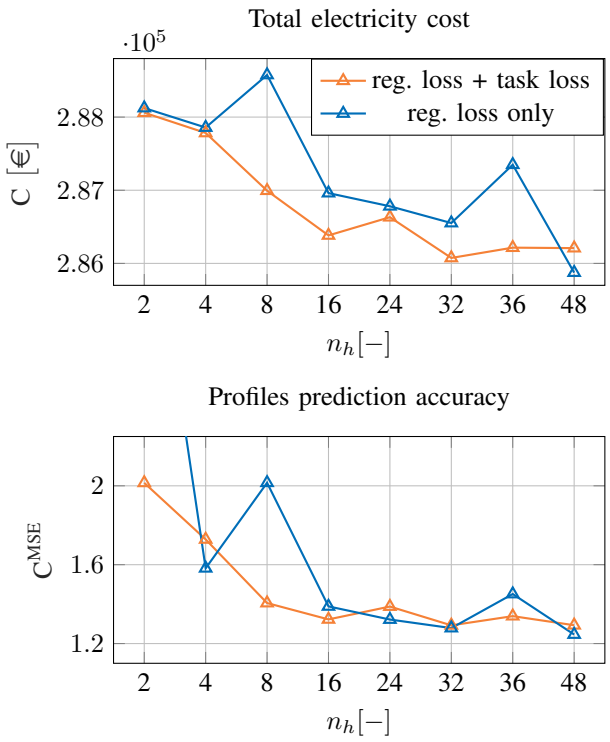

Fig. 7. Total electricity cost and profiles prediction accuracy for site 10 when the storage system parameters are perfectly known.

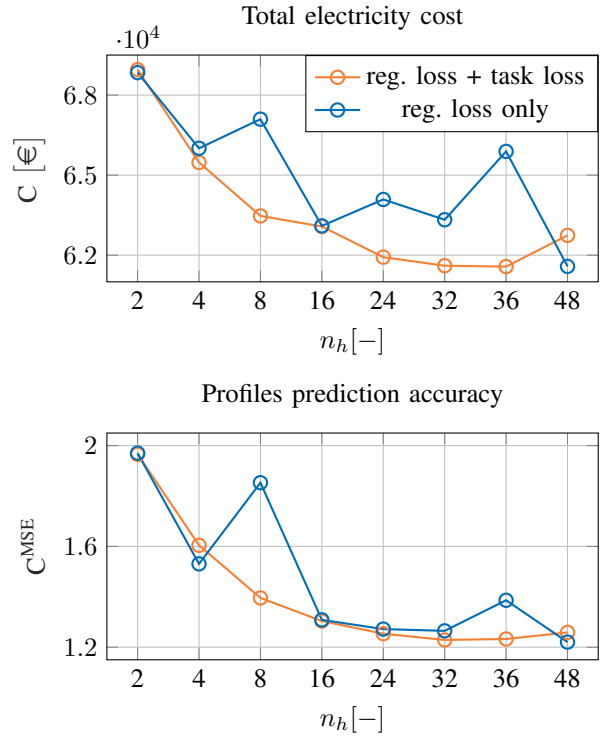

Fig. 8. Total electricity cost and profiles prediction accuracy for site 12 when the parameters of the storage system are unknown.

not available. Preliminary results were presented in [10] for the regression loss only when no model uncertainty is considered.

TABLE VII
TRANSFER LEARNING RESULTS - REGRESSION LOSS ONLY

$\mathcal{D}_1$	$\mathcal{D}_2$	TL method	C [€]	C^{MSE}
12	10	NO TL	287,227	1.250
		Only TL	292,205	12.307
		OL+TL	287,413	1.269
10	12	NO TL	62,454	1.223
		Only TL	72,601	13.655
		OL+TL	61,795	1.204

Here, we show that similar results are obtained when either the regression loss or task loss is used. We assume to have two datasets $\mathcal{D}_1$ and $\mathcal{D}_2$ related to two different microgrid systems. Dataset $\mathcal{D}_1$ is available before the controller deployment and can be used to design the EMS. Dataset $\mathcal{D}_2$ represents the microgrid system where the controller has to be implemented and is used to test the controller. We aim to show that the combination of TL and OL allows us to achieve the best performance. In other words, the lowest cost is achieved by pre-training the EMS on $\mathcal{D}_1$ and continuing the learning online on $\mathcal{D}_2$. To show this, we ran three simulations: (i) $\mathcal{D}_1$ is used for HPO, the controller is only trained online on $\mathcal{D}_2$, this is denoted as NO TL; (ii) $\mathcal{D}_1$ is used for HPO and pre-training, the controller is then deployed to $\mathcal{D}_2$ without training it online, this is denoted as Only TL; (iii) $\mathcal{D}_1$ is used for HPO and pre-training, and the controller is then deployed to $\mathcal{D}_2$ where the training continues online, denoted as OL+TL. The data samples are normalised for all the simulations using normalisation factors of $\mathcal{D}_1$. Results are shown in Table VII for the regression loss only and Table VIII for the combination of task and regression loss. The three simulations are denoted in the tables as no-TL, only-TL and OL-TL, respectively. In both cases, site 10 as $\mathcal{D}_1$ and site 12 as $\mathcal{D}_2$ and vice-versa are considered. Similar observations can be made for the two tables. Worst results are obtained in the Only TL case since the controller is not adapted online for dataset $\mathcal{D}_2$. Even though the hyperparameters have not been optimised for dataset $\mathcal{D}_2$ specifically, the online training of the controller has a beneficial effect. Hence, the best performance is obtained by combining OL and TL, again showing the OL algorithm's effectiveness. We point out that regardless of the presence or not of the pre-training phase, the computed control input is optimal with respect to the MPC cost function and satisfies the MPC constraints at each time step thanks to the presence of the last layer of the NN which is a constrained optimisation layer. Moreover, the total costs obtained in the NO TL and OL+TL cases are similar; the difference is due to the faster convergence of the OL+TL method (where the training starts with network weights that are not randomly initialised).

E. Quantile regression simulations

In this subsection, we use the method explained in Section III-G to give an online prediction of the cost over the prediction horizon. Fig. 9 compares the predicted cost (computed online as (25)), using different price quantiles, with the actual cost over the prediction horizon (calculated at the

TABLE VIII
TRANSFER LEARNING RESULTS - REGRESSION AND TASK LOSS

$\mathcal{D}_1$	$\mathcal{D}_2$	TL method	C [€]	C^{MSE}
12	10	NO TL	286,800	1.264
		Only TL	287,697	14.825
		OL+TL	286,752	1.263
10	12	NO TL	61,734	1.280
		Only TL	72,597	16.196
		OL+TL	61,319	1.243

end of the simulation as (26)). The predicted cost, over the horizon T, is computed at each time step, using the price quantile and the prediction of the power exchanged with the utility grid as given by the MPC, as:

$$\hat{C}^{\tau}_{T|t} = \sum_{k=0}^{T-1} p^{\tau}_{k|t} \hat{P}^{g}_{k|t}, \tag{25}$$

where $p^{\tau}_{k|t}$ is the τ-th quantile of the price computed at time t related to the time instant $t+k$ and $\hat{P}^{g}_{k|t}$ is the predicted power that is exchanged with the utility grid at time t over the prediction horizon T. This prediction is compared with the actual cost over the horizon $C_{T|t}$, computed at the end of the simulation as:

$$C_{T|t} = \sum_{k=0}^{T-1} p_{k|t} P^{g}_{k|t}, \tag{26}$$

$p_{k|t}$ is the true price at time $t+k$ and $P^{g}_{k|t}$ is the actual power exchanged with the utility grid at time $t+k$.

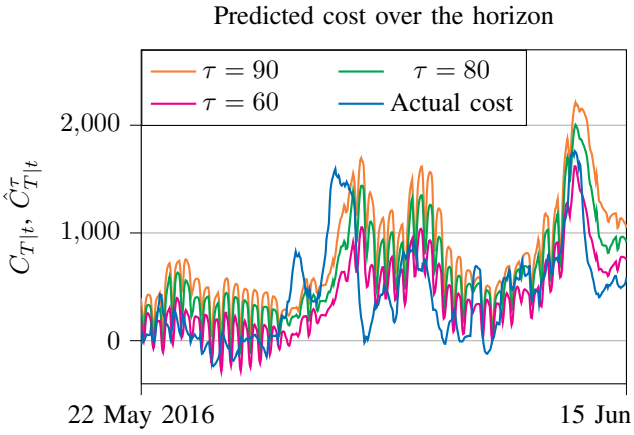

Fig. 9. True horizon cost in blue and predicted horizon cost over the horizon using different quantiles for the predicted price.

In Fig. 9, we see that the predicted cost decreases when a lower quantile is used, as expected from the definition of quantiles. Depending on the selected quantile τ, it is possible to obtain more or less conservative estimates that can be used to predict the expected cost of the microgrid operation over a future horizon with a certain confidence level. To validate these results, we counted the number of times the actual cost over the horizon is lower than the predicted cost. Results are presented in Tab. IX, where the first row shows the value of the considered quantile τ and the second row shows the percentage of times, over all the time steps in the simulations, that the actual cost is lower than the predicted cost, which is consistent with the quantile definition. For example, this information can be used for microgrid budget analysis, worst-case analysis, or risk analysis.

TABLE IX
QUANTILE REGRESSION RESULTS.

τ	60	80	90
$C_{T\|t} \leq \hat{C}^{\tau}_{T\|t}$	72%	81%	86%

V. CONCLUSION

In this paper, we propose an OL-based MPC control algorithm for microgrid EMS to learn online and adapt the predictions of the unknown profiles of price, renewable generation and load demand. O-SWA and ER algorithms have been designed for the specific online methodology to improve generalisation capabilities and avoid catastrophic forgetting. We introduced a task-specific loss so that the controller implicitly learns the uncertainty in the microgrid model, and we showed it achieves a lower total electricity cost with less complex NNs than using the regression loss only. We then showed that the combination of TL with our OL algorithm allows us to use a pre-trained NN and adapt it online to the new incoming data while improving performance. We concluded by giving an estimate of the controller memory requirement and showing that our OL algorithm can be used to estimate uncertainties of the predicted variables. As a future research direction, we would like to explore the field of online uncertainty estimation further and investigate the opportunity to leverage it in the control optimisation problem. Moreover, we will consider extending the proposed methodology to a general control framework considering non-linear plant models and different sources of uncertainty.

REFERENCES

[1] Akshay Agrawal, Shane Barratt, and Stephen Boyd. "Learning convex optimization models". In: *IEEE/CAA Journal of Automatica Sinica* 8.8 (2021), pp. 1355–1364.

[2] Akshay Agrawal et al. "Differentiable convex optimization layers". In: *Advances in neural information processing systems* 32 (2019).

[3] Akshay Agrawal et al. "Learning convex optimization control policies". In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 361–373.

[4] Verónica Álvarez, Santiago Mazuelas, and José A Lozano. "Probabilistic load forecasting based on adaptive online learning". In: *IEEE Transactions on Power Systems* 36.4 (2021), pp. 3668–3680.

[5] Brandon Amos and J Zico Kolter. "Optnet: Differentiable optimization as a layer in neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 136–145.

[6] Brandon Amos et al. "Differentiable mpc for end-to-end planning and control". In: *Advances in neural information processing systems* 31 (2018).

[7] Somil Bansal et al. "Goal-driven dynamics learning via Bayesian optimization". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 5168–5173.

[8] Vittorio Casagrande and Francesca Boem. "A novel learn-based MPC with embedded profiles prediction for microgrid energy management". In: *IFAC-PapersOnLine*. Vol. -. -. (Accepted). 2023, pp. -.

[9] Vittorio Casagrande and Francesca Boem. "Learning-based MPC using Differentiable Optimisation Layers for Microgrid Energy Management". In: *2023 European Control Conference (ECC)*. IEEE. 2023, pp. 1–6.

[10] Vittorio Casagrande et al. "An Online Learning Method for Microgrid Energy Management Control". In: *2023 31st Mediterranean Conference on Control and Automation (MED)*. IEEE. 2023, pp. 263–268.

[11] Vittorio Casagrande et al. "Resilient Microgrid Energy Management Algorithm Based on Distributed Optimization". In: *IEEE Systems Journal* ().

[12] Michelangelo Ceci et al. "Spatial autocorrelation and entropy for renewable energy forecasting". In: *Data Mining and Knowledge Discovery* 33.3 (2019), pp. 698–729.

[13] Arslan Chaudhry et al. In: *arXiv preprint arXiv:1902.10486* (2019).

[14] Steven Diamond and Stephen Boyd. "CVXPY: A Python-embedded modeling language for convex optimization". In: *Journal of Machine Learning Research* 17.83 (2016), pp. 1–5.

[15] Priya L Donti, Brandon Amos, and J Zico Kolter. "Task-based end-to-end model learning in stochastic optimization". In: *arXiv preprint arXiv:1703.04529* (2017).

[16] Ján Drgoňa et al. "Differentiable predictive control: Deep learning alternative to explicit model predictive control for unknown nonlinear systems". In: *Journal of Process Control* 116 (2022), pp. 80–92.

[17] Adam N Elmachtoub and Paul Grigas. "Smart "predict, then optimize"". In: *Management Science* 68.1 (2022), pp. 9–26.

[18] Abdellatif Elmouatamid et al. "Review of control and energy management approaches in micro-grid systems". In: *Energies* 14.1 (2020), p. 168.

[19] ENTSO-E. *Transparency platform*. URL: https://transparency.entsoe.eu. (Accessed: 14.03.2022).

[20] Mohammad Faisal et al. "Review of energy storage system technologies in microgrid applications: Issues and challenges". In: *Ieee Access* 6 (2018), pp. 35143–35164.

[21] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. Mar. 2019. DOI: 10.5281/zenodo.3828935. URL: https://github.com/Lightning-AI/lightning.

[22] Mohammad Navid Fekri et al. "Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network". In: *Applied Energy* 282 (2021), p. 116177.

[23] Aaron Ferber et al. "Mipaal: Mixed integer program as a layer". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 02. 2020, pp. 1504–1511.

[24] Helder Lopes Ferreira et al. "Characterisation of electrical energy storage technologies". In: *Energy* 53 (2013), pp. 288–298.

[25] Robert M French. "Catastrophic forgetting in connectionist networks". In: *Trends in cognitive sciences* 3.4 (1999), pp. 128–135.

[26] Christian A Hans et al. "Scenario-based model predictive operation control of islanded microgrids". In: *2015 54th IEEE conference on decision and control (CDC)* (2015), pp. 3272–3277.

[27] Lukas Hewing et al. "Learning-based model predictive control: Toward safe learning in control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3 (2020), pp. 269–296.

[28] Lion Hirth, Jonathan Mühlenpfordt, and Marisa Bulkeley. "The ENTSO-E Transparency Platform–A review of Europe's most ambitious electricity data platform". In: *Applied energy* 225 (2018), pp. 1054–1067.

[29] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[30] Pavel Izmailov et al. "Averaging weights leads to wider optima and better generalization". In: *arXiv preprint arXiv:1803.05407* (2018).

[31] Ying Ji et al. "Real-time energy management of a microgrid using deep reinforcement learning". In: *Energies* 12.12 (2019), p. 2291.

[32] Roger Koenker. *Quantile regression*. Vol. 38. Cambridge university press, 2005.

[33] James Kotary et al. "End-to-end constrained optimization learning: A survey". In: *arXiv preprint arXiv:2103.16378* (2021).

[34] Steven Spielberg Pon Kumar et al. "A deep learning architecture for predictive control". In: *IFAC-PapersOnLine* 51.18 (2018), pp. 512–517.

[35] R.H. Lasseter. "MicroGrids". In: *2002 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No.02CH37309)* 1 (2002), pp. 305–308. DOI: 10.1109/PESW.2002.985003. URL: http://ieeexplore.ieee.org/document/985003/.

[36] Adrien Le Franc et al. "EMSx: a numerical benchmark for energy management systems". In: *Energy Systems* (2021), pp. 1–27.

[37] Weirong Liu et al. "Distributed economic dispatch in microgrids based on cooperative reinforcement learning". In: *IEEE transactions on neural networks and learning systems* 29.6 (2018), pp. 2192–2203.

[38] Lennart Ljung et al. "Deep learning and system identification". In: *IFAC-PapersOnLine* 53.2 (2020), pp. 1175–1181.

[39] Jayanta Mandi and Tias Guns. "Interior Point Solving for LP-based prediction+ optimisation". In: *Advances*

in Neural Information Processing Systems 33 (2020), pp. 7272–7282.

[40] Alonso Marco et al. "Automatic LQR tuning based on Gaussian process global optimization". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 270–277.

[41] Ali Mesbah. "Stochastic model predictive control with active uncertainty learning: A survey on dual control". In: *Annual Reviews in Control* 45 (2018), pp. 107–117.

[42] Mehdi Motevasel and Ali Reza Seifi. "Expert energy management of a micro-grid considering wind energy uncertainty". In: *Energy Conversion and Management* 83 (2014), pp. 58–72.

[43] Brendan O'Donoghue et al. "Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding". In: *Journal of Optimization Theory and Applications* 169.3 (June 2016), pp. 1042–1068. URL: http://stanford.edu/~boyd/papers/scs.html.

[44] Daniel E. Olivares et al. "Trends in microgrid control". In: *IEEE Transactions on Smart Grid* 5 (4 2014), pp. 1905–1919. ISSN: 19493053. DOI: 10.1109/TSG.2013.2295514.

[45] Thomas Parisini and Riccardo Zoppoli. "A receding-horizon regulator for nonlinear systems and a neural approximation". In: *Automatica* 31.10 (1995), pp. 1443–1451.

[46] Alessandra Parisio, Evangelos Rikos, and Luigi Glielmo. "A model predictive control approach to microgrid operation optimization". In: *IEEE Transactions on Control Systems Technology* 22.5 (2014), pp. 1813–1827.

[47] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[48] Ionela Prodan and Enrico Zio. "A model predictive control framework for reliable microgrid energy management". In: *International Journal of Electrical Power & Energy Systems* 61 (2014), pp. 399–409.

[49] Naren Srivaths Raman et al. "Reinforcement learning-based home energy management system for resiliency". In: *2021 American Control Conference (ACC)*. IEEE. 2021, pp. 1358–1364.

[50] David Salinas et al. "Syne Tune: A Library for Large Scale Hyperparameter Tuning and Reproducible Research". In: *First Conference on Automated Machine Learning (Main Track)*. 2022. URL: https://openreview.net/forum?id=BVeGJ-THIg9.

[51] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25 (2012).

[52] Bharatkumar V Solanki et al. "Including smart loads for optimal demand response in integrated energy management systems for isolated microgrids". In: *IEEE Transactions on Smart Grid* 8.4 (2015), pp. 1739–1748.

[53] Shahid Ullah et al. "The Current State of Distributed Renewable Generation, Challenges of Interconnection and Opportunities for Energy Conversion based DC Microgrids". In: *Journal of Cleaner Production* (2020).

[54] Ganesh Kumar Venayagamoorthy et al. "Dynamic energy management system for a smart microgrid". In: *IEEE transactions on neural networks and learning systems* 27.8 (2016), pp. 1643–1656.

[55] Jeffrey S Vitter. "Random sampling with a reservoir". In: *ACM Transactions on Mathematical Software (TOMS)* 11.1 (1985), pp. 37–57.

[56] Leandro Von Krannichfeldt, Yi Wang, and Gabriela Hug. "Online ensemble learning for load forecasting". In: *IEEE Transactions on Power Systems* 36.1 (2020), pp. 545–548.

[57] Kim P Wabersich and Melanie N Zeilinger. "Safe exploration of nonlinear dynamical systems: A predictive safety filter for reinforcement learning". In: *arXiv preprint arXiv:1812.05506* (2018).

[58] Tiancai Wang, Xing He, and Ting Deng. "Neural networks for power management optimal strategy in hybrid microgrid". In: *Neural Computing and Applications* 31.7 (2019), pp. 2635–2647.

[59] Yeliz Yoldaş et al. "Enhancing smart grid with microgrids: Challenges and opportunities". In: *Renewable and Sustainable Energy Reviews* 72 (2017), pp. 205–214.

[60] Fuzhen Zhuang et al. "A comprehensive survey on transfer learning". In: *Proceedings of the IEEE* 109.1 (2020), pp. 43–76.

[61] Muhammad Fahad Zia, Elhoussin Elbouchikhi, and Mohamed Benbouzid. "Microgrids energy management systems: A critical review on methods, solutions, and prospects". In: *Applied energy* 222 (2018), pp. 1033–1055.