Contents lists available at ScienceDirect

Energy and AI

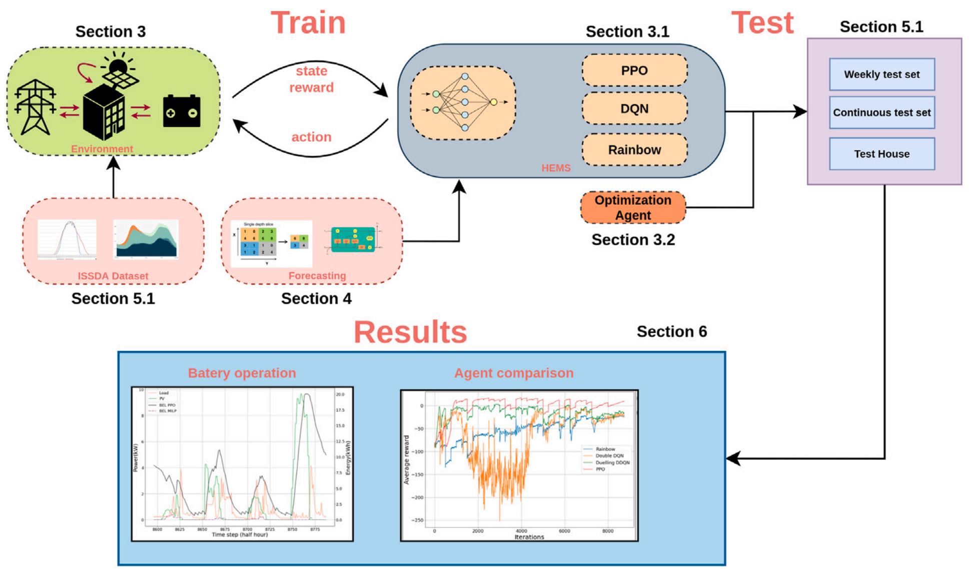
journal homepage: www.elsevier.com/locate/egyai

# Optimization of a photovoltaic-battery system using deep reinforcement learning and load forecasting

António Corte Real [a], G. Pontes Luz [b,*], J.M.C. Sousa [a], M.C. Brito [b], S.M. Vieira [a]

[a] IDMEC, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, 1049-001, Portugal
[b] Instituto Dom Luiz, Faculdade de Ciências, Universidade de Lisboa, Campo Grande, 1749-016, Lisboa, Portugal

## GRAPHICAL ABSTRACT



## HIGHLIGHTS

- A novel Home Energy Management System for a battery is proposed coupling Deep Reinforcement Learning with load forecasting using artificial neural networks.
- Reinforcement Learning agents can reach a 35% reduction in electricity bill when compared to standard model-based agents.
- Low execution time, below 10 s for 7 day solution, makes it ideal for operational implementation.

## ARTICLE INFO

*Keywords:*
Energy management system
Deep reinforcement learning
Demand-side management
Load forecasting
Deep learning

## ABSTRACT

Home Energy Management Systems (HEMS) are increasingly relevant for demand-side management at the residential level by collecting data (energy, weather, electricity prices) and controlling home appliances or storage systems. This control can be performed with classical models that find optimal solutions, with high real-time computational cost, or data-driven approaches, like Reinforcement Learning, that find good and flexible solutions, but depend on the availability of load and generation data and demand high computational resources

* Corresponding author.
*E-mail address:* gpluz@fc.ul.pt (G.P. Luz).

https://doi.org/10.1016/j.egyai.2024.100347
Received 24 October 2023; Received in revised form 12 January 2024; Accepted 23 January 2024
Available online 2 February 2024

Energy storage systems

for training. In this work, a novel HEMS is proposed for the optimization of an electric battery operation in a real, online and data-driven environment that integrates state-of-the-art load forecasting combining CNN and LSTM neural networks to increase the robustness of decisions. Several Reinforcement Learning agents are trained with different algorithms (Double DQN, Dueling DQN, Rainbow and Proximal Policy Optimization) in order to minimize the cost of electricity purchase and to maximize photovoltaic self-consumption for a PV-Battery residential system. Results show that the best Reinforcement Learning agent achieves a 35% reduction in total cost when compared with an optimization-based agent.

**Symbols**

**Abbreviations**

| | |
|---|---|
| PV | Photovoltaics |
| ANN | Artificial Neural Network |
| ML | Machine Learning |
| RL | Reinforcement Learning |
| MDP | Markov Decision Process |
| DQN | Deep Q-networks |
| PPO | Proximal Policy Optimization |
| MILP | Mixed Integer Linear Programming |
| CNN | Convolutional Neural Network |
| LSTM | Long Short-Term Memory |
| HEMS | Home Energy Management System |
| BEL | Battery Energy Level |

**Variables**

| | |
|---|---|
| $E_t^g$ | Energy imported from the grid (kWh) |
| $E_t^b$ | Energy from the battery to the load (kWh) |
| $E_t^{PV}$ | Energy from the PV system to the load (kWh) |
| $L_t$ | Load (kWh) |
| $B_t$ | Battery energy level (kWh) |
| $\hat{L}_t$ | Load forecast (kWh) |
| $L_t^{N+}$ | Positive net load (kWh) |
| $L_t^{N-}$ | Negative net load (kWh) |
| $\Delta B_t^+$ | Positive variation in the BEL (kWh) |
| $\Delta B_t^-$ | Negative variation in the BEL (kWh) |
| $E_t^{inG}$ | Energy coming from the grid to charge the battery (kWh) |
| $E_t^{inPV}$ | Energy coming from the PV system to charge the battery (kWh) |
| $E_t^{outL}$ | Energy discharged by the battery to meet local demand (kWh) |
| $E_t^{outX}$ | Energy discharged by the battery to be exported (kWh) |
| $y_t^{ch}$ | charging binary variable |
| $y_t^{dch}$ | discharging binary variable |
| $\delta$ | Energy deficit or excess (kWh) |

**Parameters**

| | |
|---|---|
| $H$ | Planing horizon |
| $p_t^b$ | grid electricity buying price (€/kWh) |
| $c_t^s$ | electricity selling price |
| $t_s$, $t_e$ | first and last timesteps of the scheduling horizon |
| $C^{ch}$ | Charging rate (kW) |
| $C^{dch}$ | Discharging rate (kW) |
| $\eta^{ch}$ | Charging efficiency |
| $\eta^{ch}$ | Discharging efficiency |
| $i$ | Minimum incremental value of battery power |
| $dh$ | Energy to power conversion |
| $B^{max}$ | Maximum Battery energy level (kWh) |

**Reinforcement Learning**

| | |
|---|---|
| $s_t$ | State at timestep $t$ |
| $u_t$ | Action at timestep $t$ (MDP and MILP) |
| $r_t$ | Reward at timestep $t$ |
| $V^\pi$ | State value function under policy $\pi$ |
| $Q^\pi$ | State–action value function under policy $\pi$ |
| $\gamma$ | MDP discount factor |
| $p(\cdot)$ | transition probability |
| $\pi_\theta$ | policy parameterized by $\theta$ |
| $\alpha$ | Learning rate |

## 1. Introduction

*Home Energy Management Systems* (HEMS) are comprised by software and hardware components, that are deployed in residential consumer homes and buildings in order to improve efficiency in energy use [1]. They integrate sensor components performing several tasks: monitor home appliances' operations, environmental variables inside buildings or receive external signals while also managing load or generation data. They integrate not only actuator components that control home appliances or storage systems after receiving commands, but also components for managing servers, databases and web applications as well [2].

An important part of HEMS is intelligence and decision making capabilities which allow advanced load and generation management strategies. These strategies may take solutions provided by optimization or machine learning algorithms that take into consideration external information on electricity prices, forecasts on future load or renewable generation among many other pieces of information [3].

HEMS are instrumental if energy users are expected to assume a more relevant role in the management of energy systems and evolve from passive consumers into *prosumers* that not only produce their own energy but can also control their load. Towards this end, Demand Side Management (DSM) is a portfolio of measures that aims to improve the energy system performance at the side of demand, i.e, perform control of home appliances (eg. dish washers, heat pumps), electrical storage systems or electrical vehicle charging, in order to satisfy grid constraints, maintain user comfort or to reduce the user's electricity bill. This last objective is the most relevant for residential prosumers that, on one hand, may be subjected to variable electricity tariffs and, on the other, possess renewable energy generation which can be used in the most efficient way. The integrated management of these two sources of electricity (grid and own renewable energy generation) allows for more complex energy management strategies that will potentially have a positive effect on the bill reduction objective but also other possible objectives such as comfort or self-sufficiency maximization.

### 1.1. Demand response

Demand Response programs is a sub-category of DSM and can be defined as the change in the electricity usage by consumers from their normal patterns, in response to changes in the price of electricity over time with the goal of lowering the energy costs [4].

Demand Response programs can be classified into two broad categories: *incentive-based* and *price-based* [5]. In Incentive-based, consumers are awarded incentives for changing their electricity demand behaviors. This may involve letting utilities directly shut-down appliances, but also responding to calls in order to curtail load due to emergencies or system contingencies. In price-based programs, consumers are charged with different electricity prices structures. There are several different schemes such as *Time-of-Use*, in which the electricity price depends on the moment in which electricity is used. This scheme reflects the variations of electricity cost with time. There are three different periods, peak, off-peak and shoulder-peak. A variation of Time-of-Use is *Critical Peak Pricing* in which prices are increased for moments in which the reliability of the power systems is jeopardized. Another important scheme is *Real Time Pricing* in which prices vary hourly.

Information on electricity prices, provided by market operators, can be integrated in HEMS and directly feed data-driven optimization or machine learning algorithms that inform decision-making concerning load management. In this way, storage systems can perform arbitrage by charging electricity when retail prices are low and discharging when they are high [6].

### 1.2. Self-consumption maximization

Solar Photovoltaic (PV) electricity generation is one of the most relevant renewable energy sources for accelerating the energy transition in Europe. as a whole. The combined solar PV in EU is expected to reach 1 TW of installed capacity by 2030 [7].

Among the most relevant characteristics of solar PV technologies lies its *scalability*, i.e., PV systems are economically and technically viable at different scales, ranging from small kW-sized residential systems to multi-MW utility sized centralized plants. Different PV system configurations also correspond to different remuneration schemes. Utility sized systems will primarily sell electricity whether in wholesale markets or through Power Purchase Agreements, while smaller systems will typically follow self-consumption schemes with the objective of consuming locally produced electricity and reduce the electricity bill. At the local level, whether residential, commercial or industrial, solar PV systems are primarily installed following this scheme and, since excess energy is poorly remunerated, PV system owners have a strong incentive to increase the share of local produced energy that is locally consumed [8] in order to offset energy imports and exports of electricity.

But, for residential consumers, there is a time lag between generation peak (at midday) and load peak (early evening) [8,9]. Therefore, to maximize PV self-consumption, one can schedule appliance usage during high energy generation periods or employ storage systems to absorb excess energy and use it later when needed.

In this case, historical and forecasted environmental variables like irradiance, temperature or future PV generation, can also be integrated in HEMS in order to perform decision-making.

### 1.3. Storage systems

With the increase in PV penetration in energy systems, storage systems are expected to gain relevance [10]. However, if using storage systems in coordination with solar PV systems seems feasible from a technical point of view, there are some economic bottlenecks that prevent PV-Battery systems from wide spread adoption. Foles et al. [11] analyze the viability of different solar PV configurations in Portugal and conclude that, although PV-battery configurations are not profitable, it is expected that, in the future, lowering costs in battery technology will make these configurations profitable. A similar conclusion is stressed by Barbour and González [9] that highlight a consensus in literature, pointing that batteries are currently less economic than PV-only systems, but that they may be economically viable if solar generation is unrewarded, wholesale prices are used and battery costs decrease. However, Crespo [12], point that increases in raw materials create uncertainty in these claims. One recent trend is that the enormous increases in wholesale and retail electricity prices due to energy crisis and the Ukraine war may render PV-Battery systems much more economically viable and appealing.

#### 1.3.1. Energy management with electric storage batteries

Storage system management falls under the DSM problems umbrella and have been typically solved with rule-based algorithms [13], classic optimization models or metaheuristics [14], i.e., model-based approaches [5]. However, applications to real use cases with these approaches imply developing complex optimization models that are expensive to solve. Moreover, energy management systems must leverage the availability of home energy data such as load and renewable generation time-series while dealing with the uncertainty in this data, a task that can be expensive for model-based approaches [15]. Machine Learning models and, specially, Reinforcement Learning (RL), have recently gained relevance in HEMS for control and optimization because they can face those challenges and because its mode of operation apply naturally to real online use-cases in which the data needed to derive control actions for battery management is streamed to HEMS that is able to perform continuous adaptation to variability in data.

In order to deal with uncertainty in electrical load and PV generation, RL models may be coupled with forecasting models. The most frequently and widely used time series forecasting models can be divided in three subcategories: statistical, Machine Learning (Artificial Neural Networks (ANN) [16], Support Vector Machines [17], Gaussian Processes) and hybrid models that combine several models [18]. But, the use of ANN-based models for time-series forecasting have gained increased relevance in the last years due to their systematic better performance against other non-ANN models, especially in scenarios for which large amounts of data are available. The current state-of-the-art of time-series forecasting models mainly use Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) and attention techniques [16,19].

### 1.4. Review of applications of deep RL to battery energy management

Vázquez-Canteli and Nagy [15] review a wide range of RL applications to demand response up until 2019. The authors also focus, specifically, on scheduling of electrical storage systems coupled with distributed RES generation. Most of the applications reviewed use Q-Learning or value-function methods and focus on optimizing the costs of consuming electricity, user comfort and minimizing electricity imports from the grid. More recent studies perform battery energy management using deep RL coupled with ANN-based forecasts for electricity prices, weather variables, load or renewable generation.

Harrold et al. [20] use the Rainbow algorithm [21] to control a battery in a microgrid taking into account load demand, solar and wind energy sources with the objective of maximizing energy cost savings. A dynamic energy pricing scheme based on the real wholesale energy market is used. A regression ANN is used to predict the demand, price, PV and wind energy generations of the following time-step. Results show that the forecast values improve the performance of the RL algorithms when compared to the cases in which they are not used. Several RL algorithms are compared against each other, where Rainbow proved to perform better.

Dorokhova et al. [22] develops an application of RL to electric vehicle (EV) charging control problem. The energy system consists of an utility grid, building load, PV generation and a single EV. Three different types of action spaces are used and, as such, three different RL algorithms are used and compared. The objective is to maximize the self-consumption of PV energy and authors conclude that the RL algorithms prove their ability in the context of the problem, although they have a considerable varying performance from episode to episode. It is also pointed out that the RL algorithms are a great promise for real-time applications due to their short execution times, when compared to Model Predictive Control algorithms.

Lu and Hong [23] propose a real-time incentive-based demand response algorithm for smart grid systems, where a forecasting model based on an ANN is used to predict the unknown prices and energy demands and to minimize future uncertainties. The input data contains maximally correlated near-term and long-term historical data. Results show that the trend in the forecast results is quite similar to the actual results.

### 1.5. Review of load forecasting using ANN

The current state-of-the-art for time-series forecasting models mainly use CNN, LSTM and attention techniques [19] but, recently, hybrid models, using two of these techniques have also been used.

Amarasinghe et al. [24] develop a CNN model to perform the energy load forecasting at an individual building level. This method uses convolutions on historical loads where the output is then fed to fully connected layers. The dataset used consists of four years of electrical load data for a single residential customer, where 3 years were used for training and one was used for testing. The CNN performs a forecast for the next 60 h while being fed the previous 60 h. For comparison, the results were compared against an ANN, SVM, LSTM and a factored restricted Boltzmann machine. It was noticed that the results did not vary much across the CNN, LSTM and the factored restricted Boltzmann machine architectures, hence concluding that CNN is a viable candidate for producing accurate load forecasts.

Zhang et al. [25] use a LSTM-based Recurrent Neural (RNN) to perform medium and short-term electric load forecasting. Since these models are developed for general time-series forecasting, they are tested in different datasets. The first dataset is a time series describing the monthly total number of passengers of the international airline totalizing 144 observations for 12 years. This dataset is helpful to examine the performance of the model in forecasting short time series with multiplicative seasonal patterns. The second dataset, represents the load of a building with 15 min resolution. For both datasets, the LSTM model is compared against a seasonal autoregressive integrated moving average model (SARIMA), a nonlinear autoregressive neural network model (NARX), a support vector regression model (SVR) and a feed-forward neural network model. For the first case, only the SARIMA model has a slightly better performance when compared to the LSTM model and for the second case the LSTM model is considerably much better than all the other models. With this, it is concluded that LSTM models are capable of forecasting complex univariate electric load time series with strong non-stationarity and non-seasonality, clearly outperforming traditional forecasting methods for short term electric load forecasting problems.

Lin et al. [26] propose a dual-stage attention based LSTM (DA-QLSTM) network for short-term zonal load probabilistic forecasting. The first stage is a feature attention based encoder responsible for calculating the impacts of the input features and the electricity load at each time step, through their correlation. The most relevant inputs can be adaptively selected then. In the second stage, there is a temporal attention based decoder responsible for mining the time dependencies. After the two stages, an LSTM model receives the attention results and probabilistic forecasts are obtained through a pinball loss function. The dataset used is comprised by five years worth of hourly electricity load and temperature. This model is compared against several other state-of-the-art load forecasting methods but the proposed model obtains the best results.

Kim and Cho [27] develop a CNN-LSTM to predict residential load. This neural network can extract spatial and temporal features which are responsible for its remarkable accuracy. The CNN layer extracts the features between the several variables that affect the load while the LSTM layer is responsible for modeling the temporal information of the irregular trends in time series components. The neural network receives several variables that affect the electric load such as voltage, intensity and sub-metering, as well as 32 household characteristics such as date and time. The model receives pre-processed data every 60 min by the sliding window algorithm as input, in order to predict the next 60 min. The dataset has a resolution of 1 min. The performance of this model was compared against other forecasting models, such as an LSTM, a Gated Recurrent Unit (GRU), a Bi-LSTM and an Attention-LSTM models.

From the surveyed works it can be concluded that convolutional structures are suited for the extraction of features between the variables that affect the output. LSTM models are able to model long-term dependencies and correlations, being the most adequate ANN architecture to problems where temporal information is relevant. Attention mechanisms benefits reside mainly in the quantification of the impacts of the input features on the target series.

### 1.6. Contributions

All the reviewed works in Section 1.4 use Q-Learning based algorithms, an evidence that is further reinforced in [28] that point out that 53% of surveyed papers are using Q-learning based algorithms. Moreover, it indicates that dueling-DQN was applied in only one article. From that review, it can also be seen that PPO is used just in a small number of papers. However, this family of algorithms have consistently showed good results in several tasks [29]. In this work a comparison of the performance of PPO against several DQN-based algorithms and an optimization-based agent is provided and results show that it outperforms all other algorithms and that PPO is an important algorithm to train energy management agents for managing storage systems. The main contributions of this work is the development of a novel HEMS for a home battery optimization system that couples a Deep Reinforcement Learning (DRL) with state-of-the art load forecasting based on CNN-LSTM considering *recurrent dropout* [30,31] in order to minimize the user energy bill considering dynamic electricity tariffs and solar PV generation. Although there are applications of CNN-LSTM to electricity consumption forecasting, the literature review in Section 1.5, shows that no other work considers the use of the recurrent dropout technique in a CNN-LSTM architecture.

### 1.7. Organization

Fig. 1 provides an overview of the interrelations among the various components involved in this work:

- **Energy management problem:** In Section 3, the general formulation of the battery energy management problem for the HEMS is provided. This problem is then formalized as an MDP or a Mixed Integer Linear Program (MILP)
- **Load forecasting module**: In Section 4, RNN, CNN and LSTM neural networks are introduced and their applications to load forecast are detailed.
- **Implementation**: In Section 5, the details of the models implementation are provided, alongside the description of the dataset used, and the details of the training and testing steps.
- **Results**: In Sections 6, 6.3 and 7, respectively the results, the discussion and conclusions are provided.

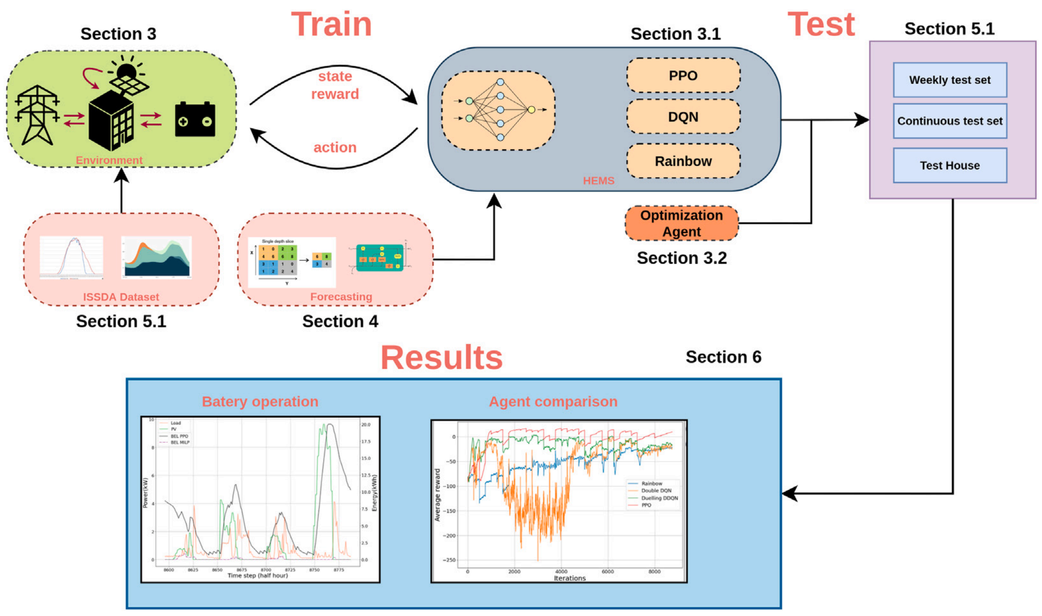


**Fig. 1.** Overview of the present work.

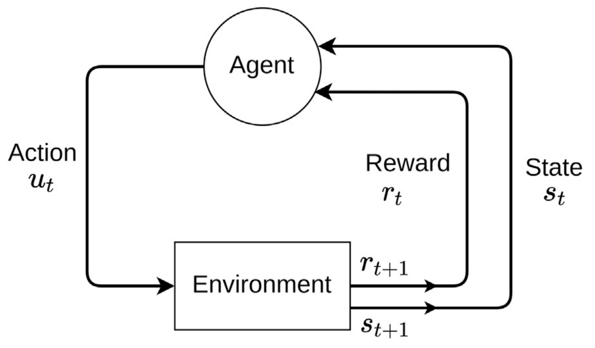


**Fig. 2.** The classic RL agent-environment interaction loop as presented in Sutton and Barto [33].

## 2. Deep reinforcement learning

RL is one of the 3 main areas of the greater ML discipline. It has gained increased attention in the last decade due to its successes in mastering board and video games [32] but also in solving control and optimization problems in engineering and energy systems [15,25]. It differs from other ML areas (supervised and unsupervised learning) in some fundamental aspects: the data used for training a RL agent is not static but dynamically collected by that same agent during the interaction with an environment (real or simulated) and, in that process, is able to build a policy so that it can act optimally in that same world. Moreover, an RL agent is not learning to label previously unseen data points from labeled training data (as in supervised learning), but it is using a reinforcement signal (reward) from which it can "discover" the best actions to take in specific states.

The classic RL interaction loop can be seen in Fig. 2 where the environment, at each timestep $t$, assumes a specific *state* $s_t$ that the agent can perceive fully or partially. Based on the *observation* of that state, the agent chooses an *action* $u_t$ based on its *policy* that is sent to the environment making it transit from $s_t$ to $s_{t+1}$. Based on the action $u_t$ taken in $s_t$, the agent receives a reward $r_{t+1}(s_t, u_t)$ (a feedback signal) that guides him in assessing whether the chosen action was good or bad [33]. The RL framework naturally models sequential decision problems in which an agent repeatedly acts on an environment and whose decisions, taken at the present state, will impact the future states and rewards. Moreover, uncertainty and stochasticity are also naturally modeled with RL: when an action is taken in state $s_t$, the next state $s_{t+1}$ is achieved according to a probability distribution.

RL problems are mathematically formalized as *Markov Decision Processes* (MDP) which are defined by the following elements: $s_t \in S$ are a set of environment states, $u_t \in A$ are the actions available to the agent, $r_t(s_t, u_t) \in R$ are the rewards received for taking the action $u_t$ in state $s_t$ and $p(s_{t+1}|s_t, u_t)$ is the probability of transitioning to $s_{t+1}$ from state $s_t$ when action $u_t$ is taken.

The agent's main task is to determine the *optimal policy* $\pi_*$, a mapping from states to actions $\pi(u_t|s_t) : S \mapsto A$ so that it maximizes the expected cumulative sum of total reward on the long run, $\mathbb{E}\left[\sum_{t=0}^{T} r_t\right]$.

In order to achieve this, two functions are of greater importance. *State-value functions* quantify the expected discounted return starting from state $s$ and following policy $\pi$ given by:

$$V^\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t | s_t = s], \quad s \in S$$

*Action-value functions* quantify the discounted expected return of taking action $u$ in $s$ and thereafter following policy $\pi$:

$$Q^\pi(s,u) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t | s_t = s, u_t = u]$$

where $0 \le \gamma \le 1$ is the discount factor, whose objective is to reduce the importance of future rewards relative to the present rewards. The *Bellman equation* is a fundamental property of value functions relating values in a given state with values on future states:

$$V^\pi(s_t) = \sum_{u} \sum_{s_{t+1}, r} p(s_{t+1}, r | s, u) \left[ r + \gamma V^\pi(s_{t+1}) \right]$$

Using these definitions, $\pi_*$ is a policy that maximizes future rewards better than any other policy which directly relates to the optimal value and state–action functions $V_*(s) = \max_\pi V^\pi(s)$ and $Q_*(s) = \max_\pi Q^\pi(s)$.

One important last result is the *Bellman optimality equation*: under the optimal policy $\pi_*$, the value of a state is the expected return when the best action is taken in that state [33]:

$$V_*(s) = \max_u Q_{\pi_*}(s,u) = \max_u \sum_{s_{t+1}, r} p(s_{t+1}, r | s, u) \left[ r + \gamma V_*(s_{t+1}) \right]$$

Almost all RL methods involve estimating whether $V^\pi(s)$ or $Q^\pi(s,u)$ and these two functions have been classically represented as matrices

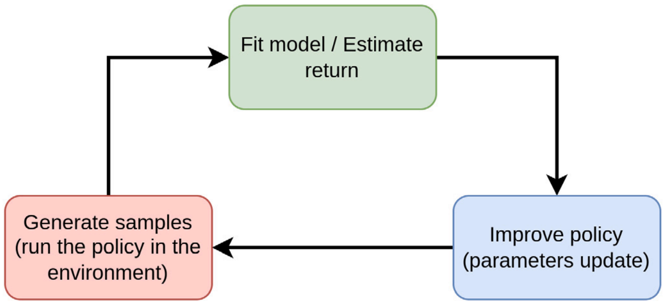

Fit model / Estimate return
Improve policy (parameters update)
Generate samples (run the policy in the environment)

**Fig. 3.** General architecture of an RL algorithm.
*Source:* [35]

which is a viable representation for small size problems. However, when facing real world problems, the matrix representations are no longer an option not only because of the possibly high state and action space dimensions but also because some problems may be represented by continuous states and actions. In these cases, function approximation becomes an option.

In the last years, with the advances in Deep Learning, ANN became a standard option to approximate any of the following components of RL: value or action-value functions $V_\theta(s)$, $Q_\theta(u,a)$, policy $\pi_\theta(u|s)$ or even the model dynamics $p_\theta(s_{t+1}|s_t,a_t)$, which become parameterized by $\theta$, i.e., the ANN weights. The function approximation capabilities of ANN paved the way for the broader area of *Deep*-RL.

### 2.1. Deep reinforcement learning algorithms

RL algorithms can be broadly divided in 3 main families: *Policy Gradient* methods in which policy $\pi_\theta$ is improved and learned directly, *Value-Based* methods in which value functions ($Q$ or $V$) are approximated and, from these, actions are taken. A third family of algorithms are *Actor–Critic* methods that mix the previous two families of algorithms: an actor uses $\pi_\theta$ to act on the environment and train a policy while a critic evaluates how good that decision was, i.e, the critic evaluates the policy [34]. Algorithms may be *on-policy* if the data used to improve a policy is generated by that same policy or *off-policy* if policies are improved using data collected with different policies. Furthermore, RL algorithms may be *model-free* if the explicit dynamics of the environment (transition probabilities $p(s_{t+1}|s_t,u_t)$) are unknown or *model-based* when the dynamics are either given or must also be estimated.

Nonetheless, RL algorithms share a common general internal architecture comprised by three main steps (Fig. 3):

- Generation of samples by running $\pi_\theta$ in the environment (data collection)
- Function fitting or approximation
- Policy improvement by performing, for example, a gradient step to update network parameters

In the remaining of this subsection we describe, in general terms, the main algorithms used in this work.

*Deep Q-networks (DQN).* DQN is one of the most popular algorithm in DRL due to its original achievements in playing Atari games [32]. It is a model-free, off-policy algorithm that can be applied to discrete action problems which is based on learning $Q_\theta(s,u)$ from collected experience of $(s,u,r,s')$ tuples. DQN is based on the classical Q-Learning algorithm [36,37] that updates the tabular $Q$ function. However, when using an approximation for $Q_\theta(s,u)$, the algorithm becomes unstable and may diverge from the $Q_{\theta*}(s,u)$. In order to address that issue, two mechanisms were introduced by the original DQN proposal [32]: *Replay buffer*, so that past experience can be reused, and the use of *target networks* in order to reduce variance.

*DQN enhancements.* In its original version, DQN is known to have an overestimation problem when estimating $Q_\theta$, which is noisy due to several possible reasons: the environment itself, non-stationarity or non-linear function approximation. The proposal of the *Double DQN* algorithm [38] aims to reduce this effect by decoupling action selection from value evaluation, which can be achieved by using two different networks for these tasks. Another enhancement relative to the original DQN is *Dueling DQN* [39] in which two streams that explicitly separate the representation of state values and state-dependent action, is proposed.

*Rainbow DQN.* The *Rainbow DQN* [21] algorithm combines the previous enhancements with 4 other extensions. *Prioritized Experience Replay* is a mechanism that allows to prioritize information-rich transitions when sampling from the replay buffer by increasing its sampling probability. *Multi-Step Learning* uses $n$-step returns to compute the targets.*Noisy Nets* are neural networks whose weights and biases are perturbed by a parametric function of noise so that the agent learns how to discard the noise entries. *Distributional*-RL approximates the distribution of returns.

*Policy gradient methods.* *Policy Gradient* methods directly approximate the policy $\pi_\theta$ by performing updates directly on its parameters $\theta$ using gradient descent. In this way, $\pi_\theta$ can model a stochastic policy from which actions can be sampled from. These methods shape $\pi_\theta$ in such a way that the probability of trajectories that lead to greater rewards is increased while the probability of worse trajectories is decreased. However, policy gradient methods suffer from high variance in the gradient estimation and a number of advances in this class of algorithms focus on reducing variance (e.g. baselines [33, Chapter 13]).

*Proximal policy optimization.* PPO [29] is an on-policy algorithm whose main idea is to improve the policy $\pi$ so that the resulting policy $\pi'$ does not differ abruptly from the previous one, potentially causing a collapse in the learning performance. PPO belongs to the family of *Trust Region* methods and constitutes an improvement relative to *Trust Region Policy Optimization* [40]. There are two PPO variants which introduce incentives so that the new policy remain close to the old policy: PPO-Penalty and PPO-Clip.

## 3. Energy management problem

In this work, the energy management problem of a grid connected residential electricity consumer which is equipped with a battery and a PV system is formulated with the objective of minimizing the cost of electricity purchase from the grid. The residential load may be supplied by the PV system, the grid or the battery system. Therefore, the problem to be solved is the optimal management of the storage system (determination of the battery charge and discharge schedule) by importing energy from the grid, charging with local PV or discharging to supply the base load. Fig. 4 shows a schematic view of the energy management problem and represent its components (PV system, grid, storage system), the energy flows between them alongside the role of the HEMS in controlling the battery system. This problem is the base decision problem that an intelligent agent integrated in a HEMS will have to solve. Therefore, it is a fundamental component of the overall workflow developed for this work as can be seen in Fig. 1. In this work, this problem is solved using Reinforcement Learning and a MILP model for benchmarking purposes.

This energy management problem is a multi timestep scheduling problem for a planning horizon $H$. At each timestep the battery must determine its charging or discharging power in order to minimize the cost of supplying the load in the long run. The total cost is given by:

$$\sum_{t=t_s}^{t_e} E_t^g p_t^b \tag{1}$$

where $E_t^g$ is the energy that is imported from the grid and $p_t^b$ is the grid's electricity tariff, $t_s$ and $t_e$ are respectively the first and last time

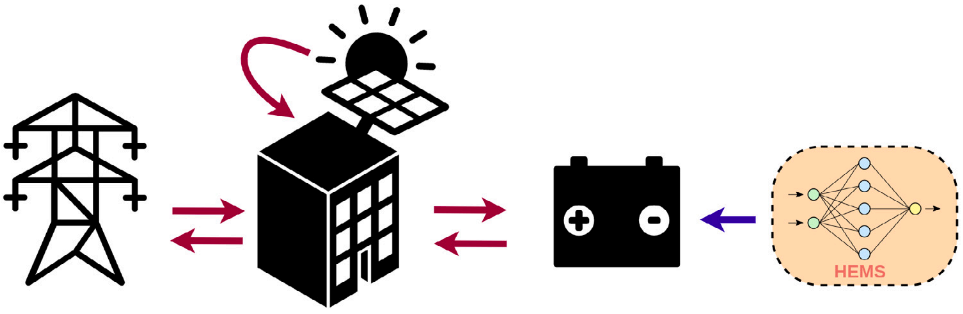


**Fig. 4.** Home Energy Management System.

steps of the considered time horizon. The house load demand $L_t$ can be met by energy coming from the battery $E_t^b$, from the grid $E_t^g$ and from the PV panels $E_t^{PV}$, every time step, thus:

$$E_t^g + E_t^b + E_t^{PV} \geq L_t \tag{2}$$

At each timestep, the power injected or extracted from the battery $u_t$ is constrained by its charging and discharging limits $C^{ch}$ and $C^{dch}$, that we assume to be symmetric ($C^{dch} = -C^{ch}$). Negative values of $u_t$ represent discharging rates and positive values represent charging rates (in kW):

$$-C^{ch} \leq u_t \leq C^{ch} \tag{3}$$

At each timestep, the Battery Energy Level (BEL), $B_t$, is limited by its maximum capacity, $B^{max}$:

$$0 \leq B_t \leq B^{max} \tag{4}$$

And the BEL update between timesteps is given by:

$$B_{t+1} = \begin{cases} B_t + \eta u_t dh, u_t > 0 \\ B_t - dh\frac{|u_t|}{\eta}, u_t < 0 \\ B_t, u_t = 0 \end{cases} \tag{5}$$

where $\eta$ is the discharging and charging efficiency of the battery and $dh$ is a energy (kWh) to power (kW) conversion factor.

These equations are the general energy management model faced by the battery. However, depending on the optimization technique used (model-based optimization or RL) several modeling implementations must be performed. One relevant difference between these techniques is the fact that, in RL, the constraints (3) and (4) cannot be hard-coded since the agent will learn from trial and error experience by collecting data in its interaction with the environment, and will be (hopefully) guided into optimal behavior through rewards. Although the specific model will be coded in the environment, the agent will not have access to it during the training phase.

### 3.1. MDP formulation

In order to apply RL to the battery energy management problem described in Section 3 it must first be mathematically formalized as an MDP. This is a classical formalization of sequential decision making problems under uncertainty, in which actions influence not just immediate rewards but also future states and rewards. The MDP must satisfy the *Markov property* which means that the evolution of the Markov process in the future depends only on the present state and not on the history of actions, states and rewards.

The RL agent's main task is to decide, at each timestep, the battery charging or discharging power rate $u_t$ in order to minimize the costs of supplying a base load. At each timestep, the battery agent finds itself in a state $s_t$ and decides to take an action based on its policy, $u \sim \pi_\theta(u_t = u|s_t)$. As a consequence, the environment emits a reward $r_t$ and transits into the next timestep state $s_{t+1}$. In the rest of this subsection the MDP elements are detailed.

*Action space.* At each timestep, the battery can either be charged, discharged or idled. The action space is discrete and represents the charging/discharging power rates that the agent can choose from:

$$U(s) = \{-C^{ch}, -C^{ch} + i, \ldots - i, 0, i, \ldots, C^{ch} - i, C^{ch}\} \tag{6}$$

where $i$ represents the minimum incremental value between each charging/discharging level

*State space.* The state encodes the information that the agent needs to take an action. In this work, the state is comprised by a total of 8 variables at each time step $t$: the PV energy generation, $E_t^{PV}$, the base load demand, $L_t$, the BEL, $B_t$, the previous timestep battery state, $B_{t-1}$, how much the BEL decreases in each time step $t$, $\Delta B_t$, the grid's imported energy, $E_t^g$, the load forecast for the next time step $\hat{L}_{t+1}$ (this quantity is provided by the load forecasting model further detailed in Section 4), and the grid electricity tariff, $p_t^b$. The state $S$ is be given by:

$$S_t = \{E_t^{PV}, L_t, B_t, B_{t-1}, \Delta B_t, E_t^g, \hat{L}_{t+1}, p_t^b\} \tag{7}$$

*Transition dynamics.* The RL algorithms used in this work are model-free, i.e, when training a policy the agent has no access to the transition dynamics of the MDP and learns from trial and error. However, in order to implement the RL environment as a simulator, this dynamic must be assumed, i.e., the modeling equations must be implemented. According to the RL loop in Fig. 2 each time the environment receives an action $u_t$, it emits a reward $r_t$ and transits into a new state $s_{t+1}$. In this section the equations of this transition are detailed.

After having received the action $u_t$, taken by the agent, the environment verifies if the current time step is not the last one, i.e $t \neq H$. If that is not the case the following updates are performed:

$$t \leftarrow t + 1 \tag{8}$$

$$\{E_t^{PV}, L_t, p_t^b\} \leftarrow \{E_{t+1}^{PV}, L_{t+1}, p_{t+1}^b\} \tag{9}$$

$$\delta_t \leftarrow E_t^{PV} - L_t \tag{10}$$

$$B_{t-1} \leftarrow B_t \tag{11}$$

$$B_t \leftarrow B_{t+1} \quad \text{according to Eq. (5)} \tag{12}$$

$$\hat{L}_t \leftarrow \hat{L}_{t+1} \quad \text{forecasting model output} \tag{13}$$

The quantity $\delta_t = E_t^{PV} - L_t$ is a variable that tracks the deficit or excess of energy in the system. If $\delta_t > 0$ there is an excess of energy, and if $\delta_t < 0$ there is a deficit (the PV generation is not enough to supply the load).

If the action taken by the agent is discharge ($u_t < 0$), the energy that is imported from the grid, $E_t^g$, is updated with the following equations:

$$E_t^g \leftarrow \begin{cases} |\delta_t - u_t| \text{ if } B_t > 0 \text{ and } \delta_t \leq 0 \text{ and } u_t \geq \delta_t \\ 0 \text{ if } B_t > 0 \text{ and } \delta_t < 0 \text{ and } u_t < \delta_t \\ 0 \text{ if } B_t > 0 \text{ and } \delta_t \geq 0 \\ \max(|\delta_t| - |\frac{\eta B_{t-1}}{dh}|, |\frac{\eta B_t}{dh}|) \text{ if } B_t < 0 \text{ and } \delta_t \leq 0 \\ |\frac{B_t}{dh}| \text{ if } B_t < 0 \text{ and } \delta_t > 0 \end{cases} \tag{14}$$

The last two cases are applied to situations where $B_t < 0$, thus violating constraint (4). The deficit is compensated by importing energy from the grid. The fourth case takes into consideration two situations: the first one, is for cases when $B_{t-1} > 0$ and $B_t < 0$ and so, $E_t^g$ is simply the amount of energy that battery could not supply, i.e., $E_t^g = |\delta_t| - |\frac{\eta B_{t-1}}{dh}|$ if $B_{t-1} > 0$ and $B_t < 0$. In the second situation there is no need to import energy since $\delta_t > 0$, but $u_t < 0$, violating constraint (4). Therefore, $E_t^g = |\frac{B_t}{dh}|$.

If the action taken by the agent is to charge or idle ($u_t \geq 0$):

$$E_t^g = \begin{cases} u_t - \delta_t \text{ if } u_t > \delta_t \\ 0 \text{ otherwise} \end{cases} \tag{15}$$

If $u_t > 0$, and $\delta_t > 0$ but $\delta_t < u_t$, energy from the grid is imported to compensate. For cases where $u_t > 0$ but $\delta_t < 0$ the energy imported from the grid needs to compensate for both.

$\Delta B_t$ is computed according to:

$$\Delta B_t = \begin{cases} \frac{|u_t| dh}{\eta} \text{ if } u_t < 0 \text{ and } B_t \geq 0 \\ B_{t-1} \text{ if } u_t < 0 \text{ and } B_t < 0 \\ 0 \text{ if } u_t \geq 0 \end{cases} \tag{16}$$

In the second case, if $B_t < 0$, it means that the battery was completely discharged, and $B_{t-1}$ represents the energy that was available to discharge.

After performing these updates the reward can be computed.

*Reward function.* The reward function $r(s, u)$ is instrumental for the learning process since it has the objective of guiding the agent in its decision process and on the determination of its policy. The overall reward function developed in this work assumes different expressions (reward groups) according to the different types of actions whether charging, discharging or idling. The greater or lower numerical value of $r(s, a)$ depends on how satisfied are the logic conditions that model the desired behavior of the agent.

The first reward term is applied for states in which the agent decides to charge the battery ($u_t > 0$) and is given by:

$$R_{ch} = \begin{cases} \frac{u_t}{\delta_t} & \text{if (18a), (18b), (18c), (18d)} \\ \frac{u_t \eta dh}{B^{max} - B_{t-1}} & \text{if (18a), (18b), (18c), (18e)} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where the conditions are given by:

$$0 \leq B_t < B^{max} \tag{18a}$$

$$\delta_t > 0 \tag{18b}$$

$$\delta_t \geq u_t \tag{18c}$$

$$B^{max} - B_{t-1} \geq \delta_t \eta dh \tag{18d}$$

$$B^{max} - B_{t-1} < \delta_t \eta dh \tag{18e}$$

Condition (18a) defines the admissible range for BEL due to operational constraints (it is the same constraint as (4) but is here reproduced for clarity). Conditions (18b), (18c) and (18d), when verified together, translate the situation in which there is still capacity in the battery, there is also excess and the excess is greater that the charging decision. Therefore, the first case in (17) is designed for situations where the battery can be charged with all the available excess. The second case is designed for situations where the battery should charge but $u_t$ has to be smaller than $\delta_t$, otherwise the upper bound of the battery, $B^{max}$, will be surpassed since there is no available capacity for all the excess. The third case is applied when the conditions of the first two cases are violated or the battery either discharges or idles. Every case is normalized so that $R_{ch}$ varies between 0 and 1. The first case is normalized by $\delta$ in order to incentivize the agent to charge the battery as much as possible. The second case is normalized by the available capacity in the battery.

The second reward term is applied to states in which the agent decides to discharge the battery ($u_t < 0$) and is given by:

$$R_{dch} = \begin{cases} \frac{1}{1 + |u_t - \delta_t|} & \text{if (18c), (18a) (20a)} \\ \frac{u_t}{\delta_t} & \text{if (18c), (18a), (20b)} \\ 0 & \text{Otherwise} \end{cases} \tag{19}$$

$$\delta_t < 0 \tag{20a}$$

$$\delta_t < u_t \tag{20b}$$

The first case in (19) is designed to punish situations where the battery discharges more energy than what it is needed for the baseload, verifying simultaneously (20a) and (20b) (energy waste) but also situations where $u_t = \delta_t$ (discharging exactly what is needed). The second case produces the reward for the situation where the battery discharges less than what is needed (20b). The last case is used when the conditions of the first two cases are not respected or the battery either charges or idles. These cases are normalized to vary between 0 and 1.

The objective of the reward in (19) is to reward the agent when it discharges the battery at a rate as close as possible to the energy deficit, i.e $u_t \approx \delta_t$ (in this case $\delta_t < 0$). However, due to the fact that the action space is discrete but the state space is continuous the exact condition $u_t = \delta_t$ is hardly ever verified. Moreover, the reward is decreased with the distance between $u_t$ and $\delta_t$. However, it is important that all the other possible actions get a reward as well, as it is better to discharge a quantity less that $\delta_t$ than to not discharge at all. It is also acceptable (but less rewarded) to discharge a quantity greater than $\delta_t$ as it still compensates the base load demand, although it is not optimal as it is wasting energy.

The third reward term is applied whenever $u_t = 0$ (idled reward), defined as:

$$R_{idle} = \begin{cases} 1 & \text{if (18a), (22a)} \lor \text{(22b), (22c)} \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

$$\delta_t = 0 \tag{22a}$$

$$\delta_t \geq 0 \tag{22b}$$

$$B^{max} \geq B_t \geq B^{max} - \eta dhi \tag{22c}$$

There are only two situations when it may not be desirable to charge or discharge the battery: when there is no excess nor deficit ($\delta_t = 0$) or when the available battery capacity is less than the minimum charging or discharging rate $i$ so that charging any amount would violate the battery operational limits (18a).

The fourth group is the grid reward, i.e, the cost of the energy that is imported from the grid:

$$R_{grid} = -E_t^g p_t^b \tag{23}$$

At last, the total reward the agent receives at each timestep is defined as:

$$R_{total} = w_1(R_{ch} + R_{dis} + R_{idle}) + w_2 R_{grid} \tag{24}$$

The total reward is a multi-objective reward function and is divided in two terms weighted by $w_1$ and $w_2$. This aims to build a signal that, on one hand, can guide the agent on choosing its actions in order to minimize the total cost of purchasing energy to supply the base load but, on the other, can choose its course of action so that all its operational constraints are satisfied.

In Eq. (24) only one of the three components, $R_{ch}$, $R_{dis}$ and $R_{idle}$, can be positive at each timestep. By trial and error, it was found that $w_1 = 0.1$ and $w_2 = 1$ got the best results and choosing $w_2 > w_1$ highlights the fact that the agent main priority is to minimize energy cost.

After performing the transition updates and computing the total reward, the back-up controller can be applied in order to ensure the feasibility of the battery energy level $B_t$.

*Back-up controller.* In order to further enforce the operational constraints, the battery is equipped with an overrule mechanism (back-up controller) that corrects $B_t$ (state variable) each time the agent action takes the MDP into a state that does not satisfy the battery capacity limits (condition (18a)). The controller acts according to a simple predefined set of rules:

$$B_t = \begin{cases} B_{max} \text{ if } B_t > B^{max} \\ 0 \text{ if } B_t < 0 \\ B_t \text{ if (18a)} \end{cases} \tag{25}$$

In the first case, when the action leads to a state with a BEL greater than the battery's maximum capacity, the controller discharges the excess energy in the battery to the maximum allowed value $E_{max}$. In the second, when the action leads to a state with a negative BEL, the deficit is compensated by importing energy from the grid.

The back-up controller is only applied after the agent has received the reward for its action from the environment. This is important for the agent learning process since violating the battery limits translates into a neutral reward ($R = 0$). In this way, the back-up controller does not interfere with the learning process.

### 3.2. MILP formulation

In this section, the energy management problem in 3 is first formulated as a MILP. Unlike the RL approach, which learns by trial and error by acting on the MDP (environment), this is a model-based approach that determines the optimal solution that can be later used for comparing the RL and the optimization-based agent. However, in order to allow for this comparison a rolling horizon optimization algorithm was developed based on the MILP formulation.

The complete MILP formulation, adapted from [9,41], is given by:

$$\text{Minimize: } \sum_{t\in H}(c_t^b L_t^{N+} + c_t^s L_t^{N-}) \tag{26a}$$

$$\text{Subjected to } \quad 0 \le E_t^{inG} \le C^{ch} \tag{26b}$$

$$0 \le E_t^{inPV} \le C^{ch} \tag{26c}$$

$$C^{dch} \le E_t^{outL} \le 0 \tag{26d}$$

$$C^{dch} \le E_t^{outX} \le 0 \tag{26e}$$

$$0 \le B_t \le B^{max} \tag{26f}$$

$$B_t = B_{t-1} + \Delta B_t^+ + \Delta B_t^- \tag{26g}$$

$$-M y_t^{ch} \le \Delta B_t^+ \le M(1 - y_t^{dch}) \tag{26h}$$

$$-M y_t^{dch} \ge \Delta B_t^- \ge M(1 - y_t^{ch}) \tag{26i}$$

$$y_t^{dch} + y_t^{ch} = 1 \tag{26j}$$

$$E_t^{inG} + E_t^{inPV} = \frac{\Delta B_t^+}{\eta^{ch}} \frac{dt}{60} \tag{26k}$$

$$E_t^{outL} + E_t^{outX} = \eta^{dch} \Delta B_t^- \frac{dt}{60} \tag{26l}$$

$$E_t^{inG} + E_t^{inPV} \le C^{ch} \tag{26m}$$

$$E_t^{outL} + E_t^{outX} \ge C^{dch} \tag{26n}$$

$$E_t^{inPV} \le -\delta^- \tag{26o}$$

$$E_t^{outL} \ge -\delta^+ \tag{26p}$$

$$E_t^{outX} = 0 \tag{26q}$$

$$L^{N+} = \delta^+ + E_t^{inG} + E_t^{outL} \tag{26r}$$

$$L^{N-} = \delta^- + E_t^{inPV} + E_t^{outX} \tag{26s}$$

This formulation models the battery dynamics by decomposing the origin of the energy that charges the battery (grid or PV system) and the destination of the energy discharged (load or exports).

The objective of this optimization model is to minimize the cost of electricity purchase given by the sum of the cost of the energy that is imported minus the revenue of what is exported. Constraints (26b), (26c), (26d), (26e) constrain the energy fluxes between the PV system, the grid, the load and the battery by the battery maximum charging and discharging rates. Constraint (26f) guarantees that the battery operates within its capacity limits and (26g) is the update rule for BEL. Binary Constraints (26h), (26i) and (26j) guarantee that the battery is not charging and discharging simultaneously. Constraints (26k) and (26l) impose that the energy that is charged or discharged from the battery is affected by their respective efficiencies and (26m) and (26n) ensures that the total energy charged or discharged from the battery remains within the limits. (26o) ensures that the energy available for charging the battery from the PV system is always less than or equal to the available excess PV and (26p) ensures that the energy used locally must be less than or equal to the demand that is not met by the PV system. At last, (26r) and (26s) define the decision variables ($L_t^{N+}$ and $L_t^{N-}$) in terms of the energy fluxes in the system.

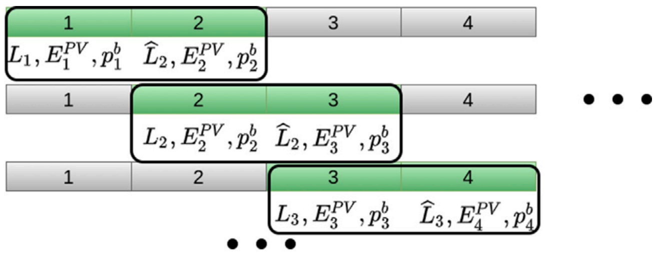

1
2
3
4
$L_1, E_1^{PV}, p_1^b$ $\hat{L}_2, E_2^{PV}, p_2^b$
$L_2, E_2^{PV}, p_2^b$ $\hat{L}_2, E_3^{PV}, p_3^b$
$L_3, E_3^{PV}, p_3^b$ $\hat{L}_3, E_4^{PV}, p_4^b$

**Fig. 5.** Online execution of the optimization-based agent.

*Optimization-based agent.* Fig. 5 shows the operation of the optimization-based developed from the model in (26). At each timestep the model is solved for a planning horizon $H = 2$ (one hour with $\Delta t = 30$ min). The agent is given the current and the next time step's state variables, except for the load demand forecast, $\hat{L}_{t+1}$ that is given by the CNN-LSTM model (see Section 4). Then, the agent saves the solution for $u_t$ determined for the first timestep and the window advances to $t + 1$. In this way is possible to develop a optimization-based agent capable of acting in an online set ups in the same way as the RL agent.

## 4. Forecasting electricity load

Another component of the battery management agents developed in this work is the residential load time-series forecasting module. Its role is to make information from the future available at present time with minimum uncertainty so that decisions taken at the present can have the future in consideration. In this work deep learning models are used since they systematically show increased performance against other models specially in scenarios for which large detasets with temporal features are available such as the present case (see Section 1.4).

The current state-of-the-art in time-series forecasting models mainly uses ANN architectures such as *Convolutional Neural Networks* (CNN), recurrent *Long Short Time Memory* (LSTM) or *Attention* mechanisms [19].

CNN is a kind of ANN aimed at processing data that is organized as a grid (time-series data can be seen as data organized in a 1D grid) and applies the linear operation of convolution, i.e., it takes the average over several input data points. Recurrent networks are a type of ANN aimed at processing sequential data such as time-series that uses parameter sharing between timesteps to model relevant relations between features. LSTM (Fig. 6) is a special type of RNN that is able to model long-term dependencies and correlations by consolidating memory units that can update an hidden state. [16].

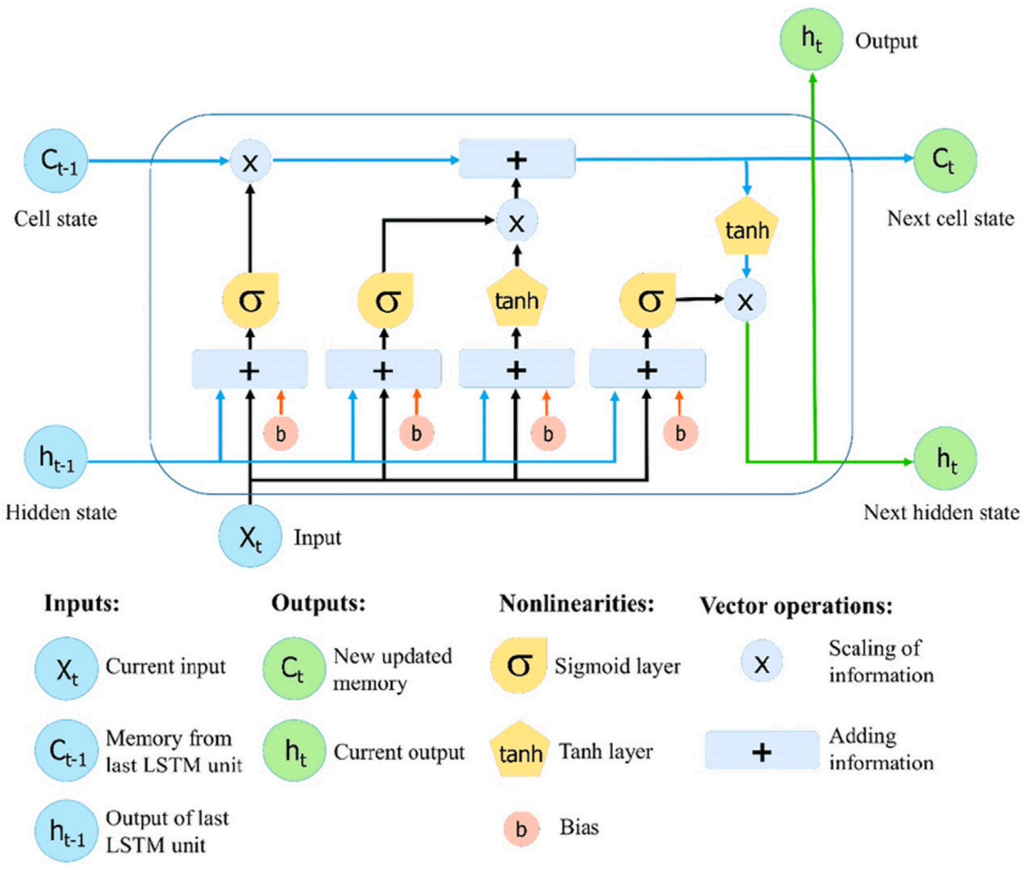


**Fig. 6.** LSTM architecture.

### 4.1. CNN-LSTM neural network

The CNN-LSTM neural network used to forecast electricity load consists of a convolutional layer followed by an LSTM layer for which the work in [27] constitutes a starting point. Such architecture is able to extract the complex features among the input features, while storing time information about important characteristics of the load, being able to predict complex and irregular trends.

However, the known overfitting problem prevents ANN from generalizing. Therefore, the *dropout* technique was used. This regularization mechanism randomly drops units of the neural network that is being trained. There are several variations on how to apply dropout to RNN and LSTM [30,31]. The recurrent dropout is the variation that will be explored. The recurrent dropout is used to regularize the recurrent weights, being applied to the recurrent connections. This technique adds a strong regularizer on the neural network weights that are responsible for learning short-term and long-term interactions.

## 5. Implementation

In the previous sections all the modeling components needed to implement and train the HEMS system depicted in Fig. 1 were detailed: the RL and optimization-based agents formulation of the energy management problem (Sections 3, 3.2) and the electricity forecasting model using a CNN-LSTM network architectures (Section 4). In this section, the implementation of these components is further detailed alongside the description of the case study dataset used for training these models.

Before proceeding, it is important to clear out that each of these components work in a different way, i.e, they are fed with different subsets of the available data, produce different outputs/solutions and are trained/solved with different algorithms. Fig. 7 presents a schematic view of the differences of each model used in this work

It is relevant to highlight that while the optimization agent directly produces a solution (an action that the battery has to take at the present timestep) for the scheduling problem, the RL agents produces a policy (a statistical distribution represented by a neural network) from which actions can be sampled.

### 5.1. Dataset and pre-processing

In the present work, whether for RL training or forecasting, the benchmark dataset on electricity load provided by the *Irish Commission for Energy Regulation* [42] was used. This dataset ranges from 2009 to 2010 with 30 min resolution and comprises data of over 5000 households and businesses. For the solar PV energy generation, PVGIS database [43] was used considering the same period for a location in Ireland with optimized solar modules slope.

*Pre-processing for RL training.* For RL training, this dataset was divided into a training and testing sets with respectively, the first 488 days and the last 37 days. The testing set is further used in two different ways:

- *Weekly Testing Set*: the testing set (37 days) is subdivided in 30 different subsets of 7 days each (some days overlap in between subsets) and the results are averaged over all subsets.
- *Continuous Testing Set*: the agents are tested in 37 consecutive days.
- *Testing House*: the agents are tested for the full data range in a different, previously unseen house, with the objective of assessing the generalization capacity.

*Pre-processing for forecasting model training.* For CNN-LSTM training, one household was chosen and the dataset was enhanced with extra features: the number of hours at each timestep (in increments of 0.5 h,

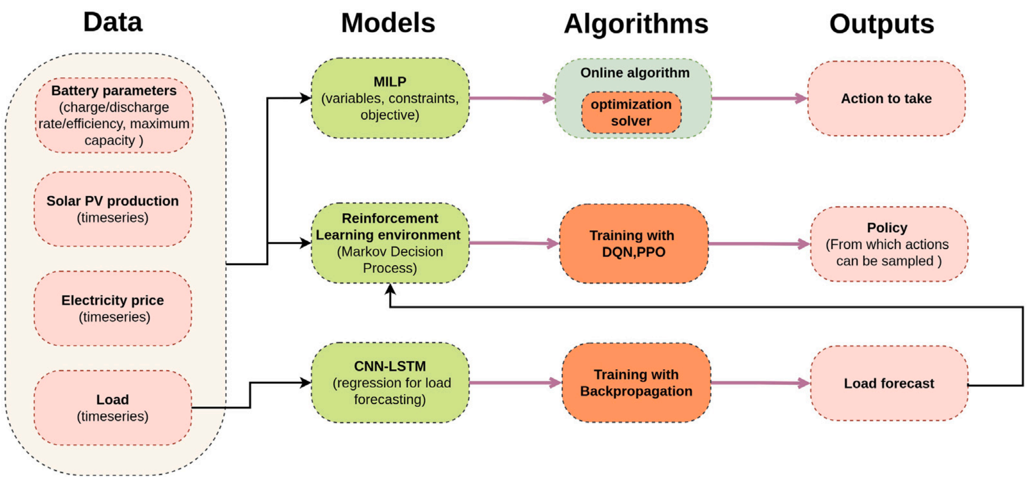

Data
Models
Algorithms
Outputs
Battery parameters (charge/discharge rate/efficiency, maximum capacity )
Solar PV production (timeseries)
Electricity price (timeseries)
Load (timeseries)
MILP (variables, constraints, objective)
Reinforcement Learning environment (Markov Decision Process)
CNN-LSTM (regression for load forecasting)
Online algorithm
optimization solver
Training with DQN,PPO
Training with Backpropagation
Action to take
Policy (From which actions can be sampled )
Load forecast

**Fig. 7.** Schematic view of the models used in the present work. Solid back lines represent data fluxes while purple arrows represent sequences of processes.

**Table 1**
Physical parameters used for the battery: grid electricity buying price values were taken from the dataset [42]; Charging/Discharging rates and efficiencies and battery capacity do not represent any specific battery model but values are in the range of common batteries found in the market.

| Parameters (unit) | Value |
|---|---|
| $H$ (Planing horizon) (days) | 7 |
| $p_t^b$ (grid electricity buying price) (€/kWh) | Tri-hourly tariff |
| 0.11 (21 h–6 h 30 m) | |
| 0.13 (6 h 30 m–15 h 30 m; 17 h 30–21 h 30 m) | |
| 0.32 (15 h 30 m–17 h 30 m) | |
| $C^{ch}$ (Charging rate) (kW) | 10 |
| $C^{dch}$ (Discharging rate) (kW) | 10 |
| $\eta^{ch}$ (Charging efficiency) | 95% |
| $\eta^{ch}$ (Discharging efficiency) | 95% |
| $i$ (Minimum incremental value of battery power) (kW) | 0.01 |
| $dh$ (Energy to power conversion) | 0.5 |
| $B^{max}$ (Maximum Battery energy level) (kWh) | 20 |

i.e, 30 min) and the weekday number (1–7). This data was normalized and was split into training (70%), testing (20%) and validation (10%) sets.

## 5.2. Energy management problem parameters

Table 1 shows the parameters defining the use case and the energy management problem

## 5.3. Reinforcement learning agents

*Hyperparameters and NN architecture.* All RL agents use the same ANN architecture (policy networks in PPO and value-functions in DQN family) of 3 hidden fully-connected layers constituted by 1024, 512 and 256 neurons, respectively with the *tanh* activation function, followed by 2 hidden fully-connected layers with 256 and 126 neurons, respectively, with the *ReLu* activation function. The output layer is a fully-connected linear layer with a single output for each valid action.

Table 2 presents the most relevant RL hyperparameters

*Reinforcement learning implementation.* The battery energy management problem was implemented as a RL environment in the *OpenAI Gym* framework [44]. The CNN-LSTM neural network was implemented using Tensorflow [45] and Keras [46]. The optimization-based agent model was developed based on the implementations of [9,41] which is implemented using the algebraic modeling language *Pyomo* [47]. The RL algorithms were available in the RLlib framework [48]. All the above mentioned methods were implemented in Python and ran on a laptop with an Intel i7-10510U CPU 1.80GHz×8 processor, 15.4 GB RAM.

### 5.3.1. Forecasting implementation

The proposed ANN architecture and hyperparameters for the electricity load forecasting CNN-LSTM model can be consulted in Table 3.

The LSTM layer has 100 units with a recurrent dropout of 30%, the first and second dense layers have 1500 and 100 units, respectively, and the last layer has 1 unit, as the output corresponds to the load of the next time step.

*CNN-LSTM training.* A sliding window algorithm was applied to use multivariate time series data as input for the LSTM. The input of the CNN-LSTM is a $147 \times 3$ matrix, corresponding to 3 days of data (30 min resolution) for 3 variables (Load $L_t$, number of hours of the time step and the current weekday) The neural network is trained on the *Stochastic Gradient Descent* based optimization algorithm *Adam* [49], with the *Mean Squared Error* (MSE), a learning rate of $10^{-4}$ and a batch size of 256. The stopping criteria was set to 80 epochs of training without loss improving on the testing data. The MSE, *Mean Absolute Error* (MAE), *Mean Absolute Percentage Error* (MAPE) and the $R^2$ regression score function were used as evaluation metrics of the model's performance, given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{L}_i - L_i)^2 \tag{27a}$$

$$MAE = \frac{\sum_{i=1}^{n}|\hat{L}_i - L_i|}{n} \tag{27b}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{\hat{L}_i - L_i}{pr_i}\right| \tag{27c}$$

$$R^2 = 1 - \frac{\sum_i(\hat{L}_i - L_i)^2}{\sum_i(L_i - \bar{L})^2} \tag{27d}$$

where $\hat{L}_i$ is the predicted load value, $L_i$ the real load value, $n$ is the number of predictions and $\bar{L}$ is the mean value of $L$ for the $n$ samples.

## 6. Results

In this section, the results of the training, whether for the forecasting neural networks or the RL agents are presented alongside the testing results.

### 6.1. Forecasting results

Table 4 summarizes the results for the three testing scenarios (see Section 5.1) showing the relevant metrics in (27).

It can be seen that the results for the Continuous Testing Set and Weekly Testing Set are quite similar, what is expected since they, in practice, account for the same days just organized in a different

**Table 2**
Hyperparameters of the RL algorithms.

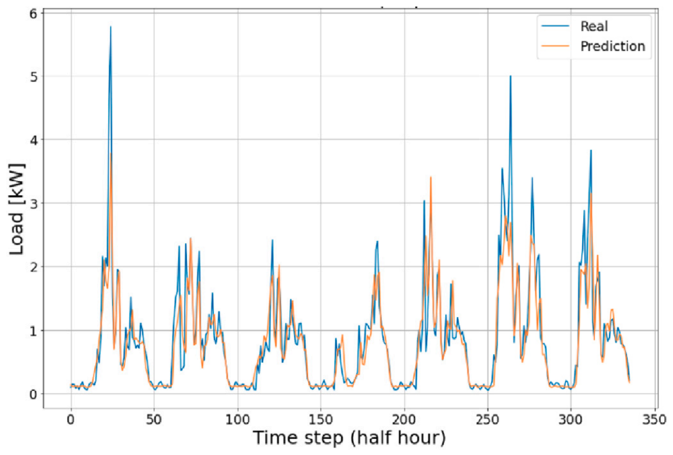
| Parameters | DDQN | Duel. DDQN | Rainbow | PPO |
|---|---|---|---|---|
| Learning rate | $5 \times 10^{-5}$ | $5 \times 10^{-5}$ | $5 \times 10^{-5}$ | $5 \times 10^{-5}$ |
| Training batch size | 1344 | 1344 | 1344 | 1344 |
| Horizon | 336 | 336 | 336 | 336 |
| Discount $\gamma$ | 0.99 | 0.99 | 0.99 | 0.99 |
| Target network update frequency | 2500 | 2500 | 8000 | – |
| Replay buffer capacity | 250 000 | 250 000 | 250 000 | – |
| Exploration $\epsilon$ | $1 \to 0.1$ | $1 \to 0.1$ | $1 \to 0.01$ | – |
| Exploration $\epsilon$ time steps | 250 000 | 250 000 | 250 000 | – |
| Prioritization exponent $\omega$ | – | 0.7 | 0.5 | – |
| Prioritization importance sampling $\beta$ | – | $0.5 \to 1$ | $0.4 \to 1$ | – |
| Noisy nets $\sigma_0$ | – | – | 0.1 | – |
| Distributional atoms | – | – | 15 | – |
| Distributional min/max values | – | – | [−33.6, 33.6] | – |
| Multi-step returns $n$ | – | – | 48 | – |
| Value function clip | – | – | – | 33.6 |

**Table 3**
Proposed CNN-LSTM architecture.

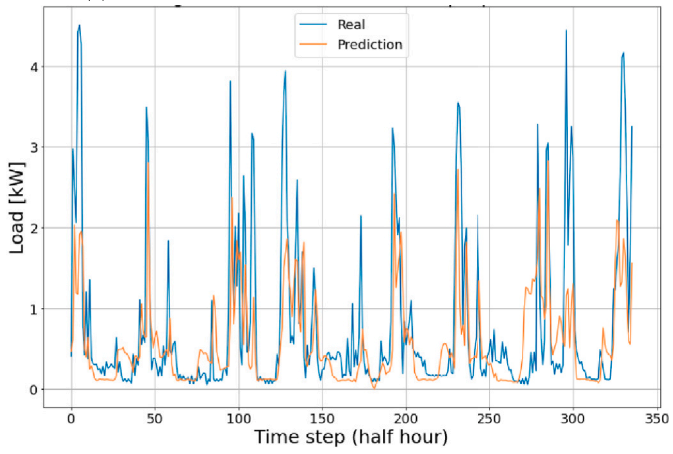
| Type | #Filter | Kernel size | Stride | Activation | #Params. |
|---|---|---|---|---|---|
| Convolution | 64 | (2, 1) | 1 | ReLu | 192 |
| Convolution | 128 | (2, 1) | 1 | ReLu | 16 512 |
| Pooling | – | (2, 1) | 1 | – | 0 |
| TimeDistributed | – | – | – | – | 91 600 |
| LSTM(100) | – | – | – | – | 80 400 |
| Dense(1500) | – | – | – | ReLu | 151 500 |
| Dense(100) | – | – | – | ReLu | 150 100 |
| Dense(1) | – | – | – | – | 101 |
| N° of params. | – | – | – | – | 490 405 |

**Table 4**
Prediction performance metrics.

| Test case | MSE | MAPE | MAE | $R^2$ |
|---|---|---|---|---|
| Weekly testing sets | 0.249 | 0.394 | 0.279 | 0.700 |
| Continuous testing set | 0.244 | 0.404 | 0.278 | 0.695 |
| Testing house | 0.558 | 0.953 | 0.453 | 0.344 |



(a) Sample of a week of predictions from the testing set.



(b) Sample of a week of predictions from the Testing House.

**Fig. 8.** The environment starts by receiving the load demand, the PV energy generation, the grid's electricity tariffs and the CNN-LSTM predictions.

way. The same is not verified for the Testing House that constitutes previously unseen data presenting, therefore, higher values of error.

In Figs. 8(a) and 8(b) random week samples from the Weekly Testing Set and from the Testing House, respectively, are presented.

As can be seen, the predictions are fairly accurate for the Weekly Testing Set showing, however, the accuracy decreases for peak values where it predicts systematically lower values when compared with observed real data. For the Testing House the performance is much lower but even though the model is capable of predicting general trends.

The results presented for the forecasting model are the baseline to apply RL. Please note that the main goal of the present work was not to train the best forecasting model but to increase the RL agent performance by providing future information. The results in Section 6.2 show that RL agents are able to manage the battery in order to reduce electricity costs in the presence of uncertainty coming from forecasting and for that they just need the general trend for the forecast.

## 6.2. *Reinforcement learning results*

In this section, the RL results for training and testing are presented alongside the comparison among the RL agents

### 6.2.1. *Training phase*

Each RL agent was trained on 35 different training sets of 7 days. The training ran for 250 iterations per each set and 4 episodes (of 7 days each) per iteration. In total, each agent is trained for 35,000 episodes, which corresponds to 8750 iterations. Fig. 9 shows the average reward evolution along iterations for the 4 agents during training. The result of each iteration is the average of the 4 episodes.

In order to develop a robust agent, the initial BEL, $B_0$, is randomly initialized to a value between 0 and $B_{max}$.

As can be observed, the PPO agent is the best performer achieving the highest average reward among the 4 agents, although, in the early training stages, it takes longer to achieve higher rewards when compared to the Double and Dueling DQN agents.

The Double DQN and the Dueling DQN agents are the RL agents that learn the fastest as after only one iteration both achieve higher rewards. The Double DQN training process revealed to be highly unstable, suffering from the previously seen "catastrophic forgetting"

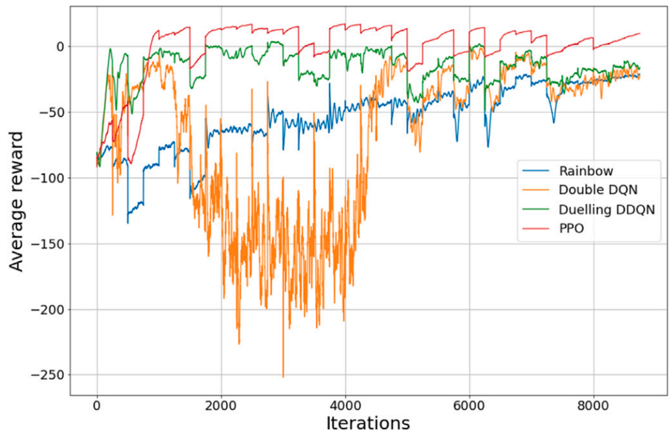
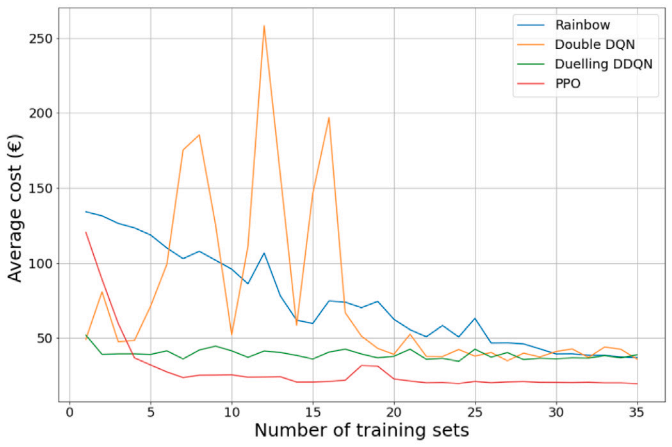


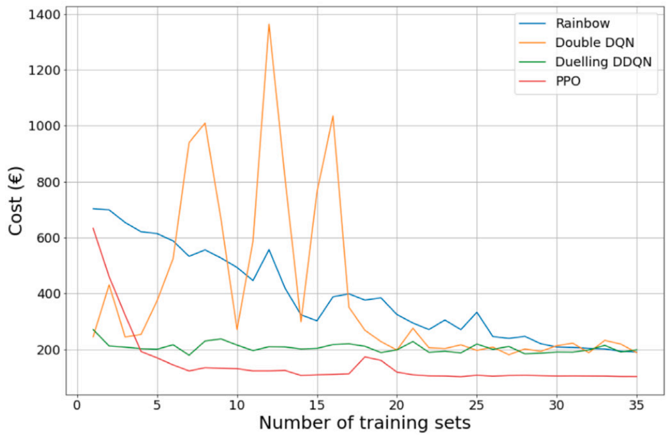
**Fig. 9.** Evolution of the average reward per iteration for the 4 RL agents.

**Table 5**
Agents' training duration in minutes.

| Algorithm | Time |
|---|---|
| Double DQN | 700 |
| Dueling DQN | 770 |
| Rainbow | 1470 |
| PPO | 1715 |



(a) Evolution of the Weekly Testing Set mean cost per weeks of training for the 4 RL agents.



(b) Evolution of the average cost of electricity purchase along the training process for the Continuous Testing Set for the four RL agents.

**Fig. 10.** Comparison between average and total cost evolution along the training process.

phenomenon [50], which happens when the agent's performance drastically drops after a period of learning. Only after the 4000 iterations the agent starts to stably learning again. Among the DQN algorithms family, the Dueling DQN shows the most stable learning process, although it still performs worse than the PPO. The Rainbow agent proved to be not only the slowest learner, but also the one that got the worst rewards.

The training duration, in minutes, of each agent is presented in Table 5.

### 6.2.2. *Testing phase*

After training the agents, these were tested so they could be compared among themselves and against the optimization-based agent implemented in Section 3.2. In the testing phase the initial conditions are deterministic and $B_0 = 0.6$ kWh, which accounts for 3% of the battery's maximum capacity ($B^{max}$). The decision to select a starting point with an almost depleted battery was motivated by the fact that agents face significant challenges when adapting to situations with limited energy availability. This choice was made to create a demanding initial scenario.

In Figs. 10(a) and 10(b) the evolution of the RL agents performance, measured by the average cost of electricity purchase, in the Weekly Testing Set and Continuous Testing Set, along the training process is presented. Each agent is trained for 250 iterations in the training set and in the end it is tested on Weekly Testing Set and in the Continuous Testing Set. The process is repeated for the 35 training sets.

From the plots it can be seen that the general trends for the four RL agents are similar for both test cases and, as such, the following analysis will be valid for both of them. By having identical trends in both situations, it can be concluded that the agents are capable of dealing with the initial conditions presented.

In the beginning, the Rainbow agent shows the worst performance but slowly improves and, by the end, it presents a similar performance when compared to the other DQN-based agents. The Double DQN agent starts by showing an acceptable performance after only the first iteration, but from the 4th until the 16th iterations, it suffers from "catastrophic forgetting" phenomenon, restarting the learning process at the 16th iteration. The Dueling DQN agent has an acceptable performance in the first iteration, but although its performance slightly improves during the remainder of the testing period its learning process is mostly stagnated. The PPO agent shows the best performance and achieves the lowest mean cost. It does not learn as fast as the Double DQN and the Dueling DQN agents, but after the 4th iteration, it consistently outperforms its counterparts.

The results obtained with the best versions of each RL agent and the optimization-based agent are presented for the 3 test cases in Table 6. For increased comparison, in Table 7 the no-battery scenarios are presented for the same test sets.

Observing the results in Tables 6 and 7, it can be concluded that the Double DQN, Dueling DQN and the Rainbow agents obtain worse results than the no-battery case. This means that there is no benefit in using this agents for energy management.

It must noted that the optimization-based agent takes into consideration more information about the next time step than the RL agents. Both the optimization-based agent and the RL agents take into account the load prediction at the next timestep, output from the CNN-LSTM, but whereas the RL agents do not take into consideration more information about the next time step, the optimization-based agent also takes the grid's electricity tariff and the PV energy as input data for the optimization. Although the optimization-based agent outputs the optimal solution every two time steps, it is evident that it is not enough for making a real impact on the final result, since these are a small subset of the whole test case.

**Table 6**
Total cost in imported energy in euros (€).

| Test case | Double DQN | Duel. DDQN | Rainbow | PPO* | MILP |
|---|---|---|---|---|---|
| Weekly testing sets | 35.6 | 38.5 | 37.4 | **20.8** | 27.6 |
| Continuous testing set | 176.2 | 183.7 | 182.5 | **106.2** | 146.6 |
| Testing house | 1925.9 | 2207.1 | 2109.5 | **1070.3** | 1653.4 |

**Table 7**
Total cost in imported energy in euros (€) without battery.

| Test case | Cost |
|---|---|
| Weekly testing sets (1 week) | 29.4 |
| Continuous testing set (37 days) | 155.3 |
| Testing house (478 days) | 1734.8 |

The PPO agent achieved the best results for every test case. The PPO agent decreased the total cost in 24.6%, 28.5% and 35.3%, respectively for the Weekly Testing Set, Continuous Testing Set and Testing House, when compared to the optimization-based agent. Moreover, the PPO agent performed the best in the Testing House, which is the most challenging test case. Comparing to the scenario with no energy management, the PPO agent achieves 29.2%, 31.6% and 38.3% cost decreases for the test cases.

In Figs. 11(a) and 11(b) a comparison of the BEL solution for the PPO (the best RL agent) and MILP agents for Weekly Testing Set (3 days) and Testing House (4 days) is presented. For simplicity of reading only $L_t$ and $E_t^{PV}$ are plotted against BEL. As can be seen, the PPO agent manages the battery in a rather expected way: it charges with excess PV electricity and discharges when there is deficit. The optimization-based agent shows a poor performance making an insufficient use of the battery.

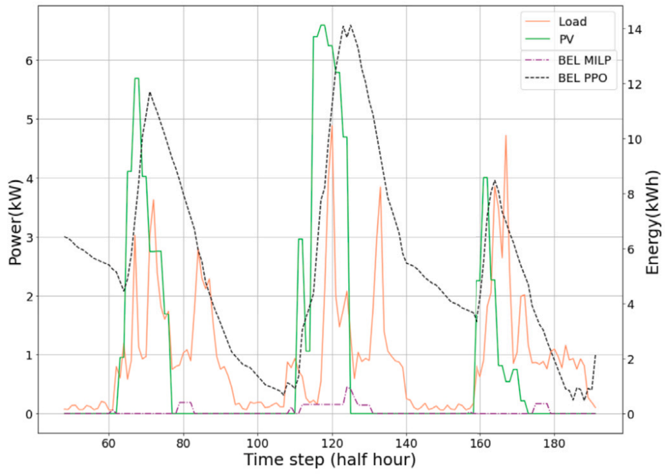
The execution time of each RL agent and the optimization-based agent is presented in Table 8. The four RL agents are much faster at solving each test case than the optimization-based agent which illustrates the time-cost benefits of RL agents when compared with classical optimization ones.

### 6.3. Discussion

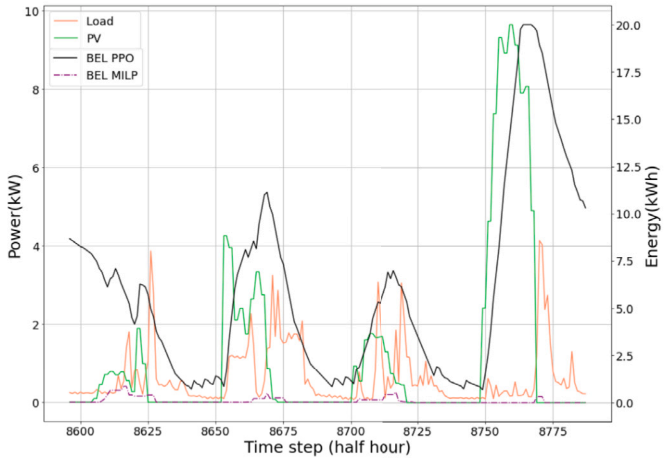
The tests made on the Testing House were a way of assessing the generalization capabilities of the trained agents when faced with previously unseen data. One immediate conclusion is that the PPO agent was the only agent able to minimize the total cost and, hence, being successful in the main energy management problem. From the execution times, it can be seen that the RL agents are much more adequate to online scenarios as they take much less time to decide on what actions to take, compared to the optimization-based agent, that needs to build and solve an optimization problem. Concerning this agent, it should be noted that its horizon is only 2 timesteps and, if this horizon would be increased, the execution times would be even larger. It must be noted that model-based optimization differs from the RL approach in an important point: while the latter takes a considerably large time and computational resources to be be trained, the former does not imply training but takes a considerable computational cost to solve the problem to optimality or even and acceptable solution. However, after the training is complete, the RL agents can solve a scenario in a much shorter time than the optimization-based agent, what is better suited for solving online scenarios, as these are expected to constitute real cases in houses where batteries are to be installed.

## 7. Conclusions

In this work, RL agents were applied to an energy management problem and a load forecasting model based on CNN-LSTM models was developed from scratch so they could be integrated in a HEMS for managing a residential battery considering PV generation and electricity tariffs. Furthermore, for comparison purposes, a battery management MILP model was also adapted into an online optimization-based agent in order to compare against the RL agents.



(a) Weekly Testing Set



(b) Testing House.

**Fig. 11.** Comparison of the Battery Energy Level between the PPO algorithm and the optimization agent.

Several RL algorithms were tested for different test cases so that a comparison could be performed in terms of the savings achieved in electricity purchase from the grid. Among the tested agents, the PPO agent is, by far, the best performer being able to achieve savings in electricity bill of 38.3% when compared with the case when there is no energy management and 35.3% when compared with the optimization-based agent. Moreover, the PPO-based agent is suitable for real-world applications due to its low execution times since it only takes 1.4 s to solve a scenario with 7 days, 3.4 s for 37 days and 44.4 s for 478 days. All the RL-based agents show the same order of magnitude for execution times. While the optimization-based agent serves as a valuable benchmark model, it falls short for real-world applications due to its significant computational inefficiency. For instance, when

**Table 8**
Execution time in seconds (s).

| Test case | Double DQN | Duel. DDQN | Rainbow | PPO | MILP |
|---|---|---|---|---|---|
| Weekly testing sets | 1.3 | 1.4 | 3.5 | 1.4 | 45.6 |
| Continuous testing set | 3.6 | 3.8 | 15.0 | 3.4 | 312.0 |
| Testing house | 45.7 | 46.9 | 193.2 | 44.4 | 4468.5 |

compared to the PPO agent, it takes approximately 100 times longer to solve scenarios like the Testing House, making it impractical in practice. DQN-based agents revealed to be unsuitable for effectively reducing the energy cost for residential consumers.

Concerning the training phase duration, the PPO agent took roughly 29 h to fully train. This was expected, as RL agents typically need a considerable amount of computational resources. This highlights a great challenge concerning the use of such agents in real-world applications.

A load forecasting model, based on a CNN-LSTM neural network, was also developed in this work with the objective of increasing the RL agents performance. This is a state-of-the-art time series forecasting neural network, which made use of the architecture that has been found to deliver the best results. The results show that the forecasting model obtained identical MSE $\sim$ 0.24, MPE $\sim$ 0.4, MAE between 0.178 and 0.279 and $R^2 \sim 0.7$ for the Weekly Testing Set and Continuous Testing Set. As expected, the model does not obtain such good results for the Testing House obtaining an MSE of 0.558, an MPE of 0.953, MAE of 0.453 and an $R^2$ of 0.344.

The present work open up several research avenues in some of the components that make part of this work. In terms of case study, future work will focus on investigating the behavior of the agents when faced with fully dynamic tariffs. In terms of the forecasting, future work will develop a longer sighted model that is able of predicting several timesteps ahead in order to increase the RL agents performance. Future work will explore the impact of mixing data from different houses during training in order to improve performance.

## CRediT authorship contribution statement

**António Corte Real:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **G. Pontes Luz:** Conceptualization, Writing – original draft, Writing – review & editing. **J.M.C. Sousa:** Conceptualization, Writing – review & editing. **M.C. Brito:** Writing – review & editing. **S.M. Vieira:** Conceptualization, Supervision, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

The energy data used in this manuscript is available (see details in the manuscript). The code used to perform experiments and analysis will be made available on request.

## Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used, sporadically, GPT-3.5 in order to improve english written language. After using it, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

## Acknowledgments

This work was funded by the Portuguese Portuguese Fundação para a Ciência e a Tecnologia (FCT) I.P./MCTES through IDMEC under LAETA, Project:
UIDB/50022/2020 (https://doi.org/10.54499/UIDB/50022/2020), UIDP/50022/2020 (https://doi.org/10.54499/UIDP/50022/2020) and through national funds (PIDDAC), through
IDL – UIDB/50019/2020 https://doi.org/10.54499/UIDB/50019/2020), UIDP/50019/2020 (https://doi.org/10.54499/UIDP/50019/2020) and LA/P/0068/2020 (https://doi.org/10.54499/LA/P/0068/2020) and the FCT Studentship 2021.06299.BD (G Pontes Luz)

## References

[1] Zandi H, Kuruganti T, Vineyard EA, Fugate D. Home energy management systems: An overview. Oak Ridge National Laboratory; 2018, URL https://www.osti.gov/biblio/1423114.

[2] Zafar U, Bayhan S, Sanfilippo A. Home energy management system concepts, configurations, and technologies for the smart grid. IEEE Access 2020;8:119271–86. http://dx.doi.org/10.1109/ACCESS.2020.3005244.

[3] Shareef H, Ahmed MS, Mohamed A, Al Hassan E. Review on home energy management system considering demand responses, smart technologies, and intelligent controllers. IEEE Access 2018;6:24498–509. http://dx.doi.org/10.1109/ACCESS.2018.2831917.

[4] Albadi MH, El-Saadany EF. Demand response in electricity markets: An overview. In: 2007 IEEE power engineering society general meeting. IEEE; 2007, p. 34–123. http://dx.doi.org/10.1109/pes.2007.385728.

[5] Jordehi AR. Optimisation of demand response in electric power systems, a review. Renew Sustain Energy Rev 2019;103:308–19. http://dx.doi.org/10.1016/j.rser.2018.12.054.

[6] Núñez F, Canca D, Arcos-Vargas Á. An assessment of European electricity arbitrage using storage systems. Energy 2022;242:122916. http://dx.doi.org/10.1016/j.energy.2021.122916.

[7] Meban B. RePower EU with solar: The 1TW EU solar pathway for 2030. 2023, https://www.solarpowereurope.org/press-releases/re-power-eu-with-solar-the-1-tw-eu-solar-pathway-for-2030. (Accessed 30 April 2023).

[8] Reis V, Almeida RH, Silva JA, Brito MC. Demand aggregation for photovoltaic self-consumption. Energy Rep 2019;5:54–61. http://dx.doi.org/10.1016/j.egyr.2018.11.002.

[9] Barbour E, González MC. Projecting battery adoption in the prosumer era. Appl Energy 2018;215:356–70. http://dx.doi.org/10.1016/j.apenergy.2018.01.056.

[10] Higinbotham T. Three battery energy storage trends for the electrification of everything. 2022, https://www.pv-magazine.com/2022/05/03/two-battery-energy-storage-trends-for-the-electrification-of-everything/. (Accessed 30 April 2023).

[11] Foles A, Fialho L, Collares-Pereira M. Techno-economic evaluation of the Portuguese PV and energy storage residential applications. Sustain Energy Technol Assess 2020;39:100686. http://dx.doi.org/10.1016/j.seta.2020.100686.

[12] Crespo CB. Developing a battery management system for self-consumption systems. [Master's thesis], Faculdade de Ciências da Universidade de Lisboa; 2022, available at http://hdl.handle.net/10451/56383.

[13] Shakeri M, Shayestegan M, Reza SS, Yahya I, Bais B, Akhtaruzzaman M, Sopian K, Amin N. Implementation of a novel home energy management system (HEMS) architecture with solar photovoltaic system as supplementary source. Renew Energy 2018;125:108–20. http://dx.doi.org/10.1016/j.renene.2018.01.114.

[14] Gholami M, Sanjari M. Multiobjective energy management in battery-integrated home energy systems. Renew Energy 2021;177:967–75. http://dx.doi.org/10.1016/j.renene.2021.05.162.

[15] Vázquez-Canteli JR, Nagy Z. Reinforcement learning for demand response: A review of algorithms and modeling techniques. Appl Energy 2019;235:1072–89. http://dx.doi.org/10.1016/j.apenergy.2018.11.002.

[16] Goodfellow I, Bengio Y, Courville A. Deep learning. MIT Press; 2016, http://www.deeplearningbook.org.

[17] Cortes C, Vapnik V. Support-vector networks. Mach Learn 1995;20(3):273–97.

[18] Lin Y, Luo H, Wang D, Guo H, Zhu K. An ensemble model based on machine learning methods and data preprocessing for short-term electric load forecasting. Energies 2017;10(8):1186. http://dx.doi.org/10.3390/en10081186.

[19] Lim B, Zohren S. Time-series forecasting with deep learning: a survey. Phil Trans R Soc A 2021;379(2194):20200209. http://dx.doi.org/10.1098/rsta.2020.0209.

[20] Harrold DJ, Cao J, Fan Z. Data-driven battery operation for energy arbitrage using rainbow deep reinforcement learning. Energy 2022;238:121958. http://dx.doi.org/10.1016/j.energy.2021.121958.

[21] Hessel M, Modayil J, Hasselt HV, Schaul T, Ostrovski G, Dabney W, et al. Rainbow: Combining improvements in deep reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence, vol. 32, no. 1. Association for the Advancement of Artificial Intelligence (AAAI); 2018, http://dx.doi.org/10.1609/aaai.v32i1.11796.

[22] Dorokhova M, Martinson Y, Ballif C, Wyrsch N. Deep reinforcement learning control of electric vehicle charging in the presence of photovoltaic generation. Appl Energy 2021;301:117504. http://dx.doi.org/10.1016/j.apenergy.2021.117504.

[23] Lu R, Hong SH. Incentive-based demand response for smart grid with reinforcement learning and deep neural network. Appl Energy 2019;236:937–49. http://dx.doi.org/10.1016/j.apenergy.2018.12.061.

[24] Amarasinghe K, Marino DL, Manic M. Deep neural networks for energy load forecasting. In: 2017 IEEE 26th international symposium on industrial electronics. ISIE, 2017, p. 1483–8. http://dx.doi.org/10.1109/ISIE.2017.8001465.

[25] Zhang Z, Zhang D, Qiu RC. Deep reinforcement learning for power system applications: An overview. CSEE J Power Energy Syst 2020;6(1):213–25. http://dx.doi.org/10.17775/CSEEJPES.2019.00920.

[26] Lin J, Ma J, Zhu J, Cui Y. Short-term load forecasting based on LSTM networks considering attention mechanism. Int J Electr Power Energy Syst 2022;137:107818. http://dx.doi.org/10.1016/j.ijepes.2021.107818.

[27] Kim T-Y, Cho S-B. Predicting residential energy consumption using CNN-LSTM neural networks. Energy 2019;182:72–81. http://dx.doi.org/10.1016/j.energy.2019.05.230.

[28] Al-Ani O, Das S. Reinforcement learning: Theory and applications in HEMS. Energies 2022;15(17):6392. http://dx.doi.org/10.3390/en15176392.

[29] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017, http://dx.doi.org/10.48550/ARXIV.1707.06347, arXiv URL https://arxiv.org/abs/1707.06347.

[30] Gal Y, Ghahramani Z. A theoretically grounded application of dropout in recurrent neural networks. 2015, http://dx.doi.org/10.48550/ARXIV.1512.05287, arXiv URL https://arxiv.org/abs/1512.05287.

[31] Moon T, Choi H, Lee H, Song I. RNNDROP: A novel dropout for RNNS in ASR. In: 2015 IEEE workshop on automatic speech recognition and understanding. ASRU, IEEE; 2015, p. 1–2. http://dx.doi.org/10.1109/asru.2015.7404775.

[32] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, et al. Human-level control through deep reinforcement learning. Nature 2015;518(7540):529–33. http://dx.doi.org/10.1038/nature14236.

[33] Sutton RS, Barto AG. Reinforcement learning. In: Adaptive computation and machine learning series. 2nd ed. Cambridge, MA: Bradford Books; 2018.

[34] Dong H, Ding Z, Zhang S. Deep reinforcement learning: fundamentals, research and applications. Cham, Switzerland: Springer Nature; 2020.

[35] Levine S. Notes from the course CS285: Deep reinforcement learning at UC berkeley. 2020, https://rail.eecs.berkeley.edu/deeprlcourse/.

[36] Watkins CJCH. Learning from delayed rewards [Ph.D. thesis], Cambridge, England: University of Cambridge; 1989.

[37] Watkins CJCH, Dayan P. Q-learning. Mach Learn 1992;8(3–4):279–92. http://dx.doi.org/10.1007/bf00992698.

[38] Hasselt H. Double Q-learning. In: Lafferty J, Williams C, Shawe-Taylor J, Zemel R, Culotta A, editors. In: Advances in neural information processing systems, vol. 23, Curran Associates, Inc.; 2010, p. 1–2.

[39] Wang Z, Schaul T, Hessel M, van Hasselt H, Lanctot M, de Freitas N. Dueling network architectures for deep reinforcement learning. 2016, arXiv:1511.06581.

[40] Schulman J, Levine S, Moritz P, Jordan MI, Abbeel P. Trust region policy optimization. 2015, http://dx.doi.org/10.48550/ARXIV.1502.05477, arXiv URL https://arxiv.org/abs/1502.05477.

[41] Barbour E. Optimal scheduling of battery energy storage systems. Energy Systems and Energy Storage Lab; 2018, URL http://www.eseslab.com/posts/blogPost_batt_schedule_optimal.

[42] Commission for Energy Regulation (CER). CER smart metering project - Electricity customer behaviour trial, 2009–2010 dataset. 2012, https://www.ucd.ie/issda/data/commissionforenergyregulationcer/.

[43] European Commission, Joint Research Centre. Photovoltaic geographical information system. 2022, https://re.jrc.ec.europa.eu/pvg_tools/en/. [Accessed 210 February 2023].

[44] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. Openai gym. 2016, http://dx.doi.org/10.48550/ARXIV.1606.01540, arXiv URL https://arxiv.org/abs/1606.01540.

[45] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. 2016, http://dx.doi.org/10.48550/ARXIV.1603.04467, arXiv URL https://arxiv.org/abs/1603.04467.

[46] Chollet F, et al. Keras. 2015, https://keras.io.

[47] Hart WE, Watson J-P, Woodruff DL. Pyomo: modeling and solving mathematical programs in Python. Math Program Comput 2011;3(3):219–60. http://dx.doi.org/10.1007/s12532-011-0026-8.

[48] Moritz P, Nishihara R, Wang S, Tumanov A, Liaw R, Liang E, et al. Ray: A distributed framework for emerging AI applications. 2017, http://dx.doi.org/10.48550/ARXIV.1712.05889, arXiv URL https://arxiv.org/abs/1712.05889.

[49] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, http://dx.doi.org/10.48550/ARXIV.1412.6980, arXiv URL https://arxiv.org/abs/1412.6980.

[50] Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu AA, et al. Overcoming catastrophic forgetting in neural networks. Proc Natl Acad Sci 2017;114(13):3521–6. http://dx.doi.org/10.1073/pnas.1611835114.