05

PHENIKAA UNIVERSITY
PHENIKAA SCHOOL OF ENGINEERING
FACULTY OF ELECTRICAL AND ELECTRONIC ENGINEERING

FINAL ASSIGNMENT REPORT

ADVANCED REINFORCEMENT LEARNING

Deep Reinforcement Learning for Smart Home Energy Management

Student: Nguyen Thi Huong Student ID: 20010741

Supervisor: PhD. Vu Hoang Dieu Electrical and Electronics Engineering

Score

Lecture 1	Lecture 2

Hanoi | 18.07.2025

Contents

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

With residential energy costs on the rise, this project explores an intelligent solution for smart home energy management. The primary objective is to minimize electricity expenses for a home featuring an HVAC system and a battery (ESS), without sacrificing occupant comfort. I implemented a Deep Reinforcement Learning (DRL) agent based on the Deep Deterministic Policy Gradient (DDPG) algorithm to achieve this. This model-free approach is ideal as it requires no prior knowledge of the building's thermal dynamics and can handle the continuous action spaces needed for precise HVAC and battery control. A key aspect of this project was training a single agent to learn a coordinated control strategy for both systems simultaneously. The performance of the trained agent, tested on real-world data, was highly effective. Beyond cost savings, the agent consistently maintained thermal comfort and demonstrated emergent, intelligent behaviors, such as aligning battery charging cycles with low-cost electricity periods. Ultimately, this project validates that DDPG can be used to create a robust and autonomous agent capable of sophisticated energy optimization in a dynamic smart home environment.

Keywords: *Deep Reinforcement Learning, Smart Home Energy Management, DDPG, HVAC Control, Energy Storage Systems (ESS), Energy Cost Minimization*

1. INTRODUCTION

1.1 Problem:

In this project, I address the challenge of high energy costs in modern smart homes. The core problem is to develop an autonomous control system that minimizes the daily electricity bill for a household equipped with two key components: a controllable Heating, Ventilation, and Air Conditioning (HVAC) system and a battery-based Energy Storage System (ESS). The system must achieve this cost reduction while simultaneously satisfying a critical constraint: maintaining the indoor temperature within a predefined comfort range for the occupants. This task is complicated by the inherent uncertainties of the real world, such as unpredictable fluctuations in solar energy generation, varying electricity prices, and unknown thermal properties of the building.

1.2 Motivation:

My motivation for this project stems from the significant real-world impact of residential energy consumption. HVAC systems are among the most power-hungry appliances in a typical home, often responsible for 40% or more of the total energy bill. With the rise of smart grids and dynamic pricing tariffs, there is a clear financial incentive to manage this consumption intelligently. An effective energy management system not only provides direct economic benefits to the homeowner but also supports the broader goal of grid stability by reducing demand during peak hours. I was motivated by the challenge of creating a truly "smart" agent that could learn to navigate these complexities on its own, without being explicitly programmed with rules.

1.3 My Idea:

My core idea is to leverage the power of Deep Reinforcement Learning (DRL) to create an intelligent agent capable of solving this complex optimization problem. I selected the Deep Deterministic Policy Gradient (DDPG) algorithm as the foundation of my approach. I chose DDPG specifically because of its ability to operate effectively in environments with continuous state and action spaces. This is crucial for my problem, as the agent needs to decide on precise, continuous values for battery power (e.g., charge at 1.5 kW) and HVAC power (e.g., run at 0.8 kW), rather than discrete on/off levels.

The key innovation of my implementation is to train a single, unified agent to make joint decisions for both the ESS and the HVAC system simultaneously. Instead of treating them as separate problems, my agent learns the intricate interplay between them. For instance, it can learn to use the energy stored in the battery to power the HVAC when grid electricity is expensive, a sophisticated strategy that is difficult to hard-code but can be discovered through DRL.

To provide a clear context for this project, the architecture of the simulated smart home environment is illustrated in Figure 1 below. This diagram outlines the essential components and the interactions between them, which form the basis of the Markov Decision Process (MDP) that our reinforcement learning agent will navigate.

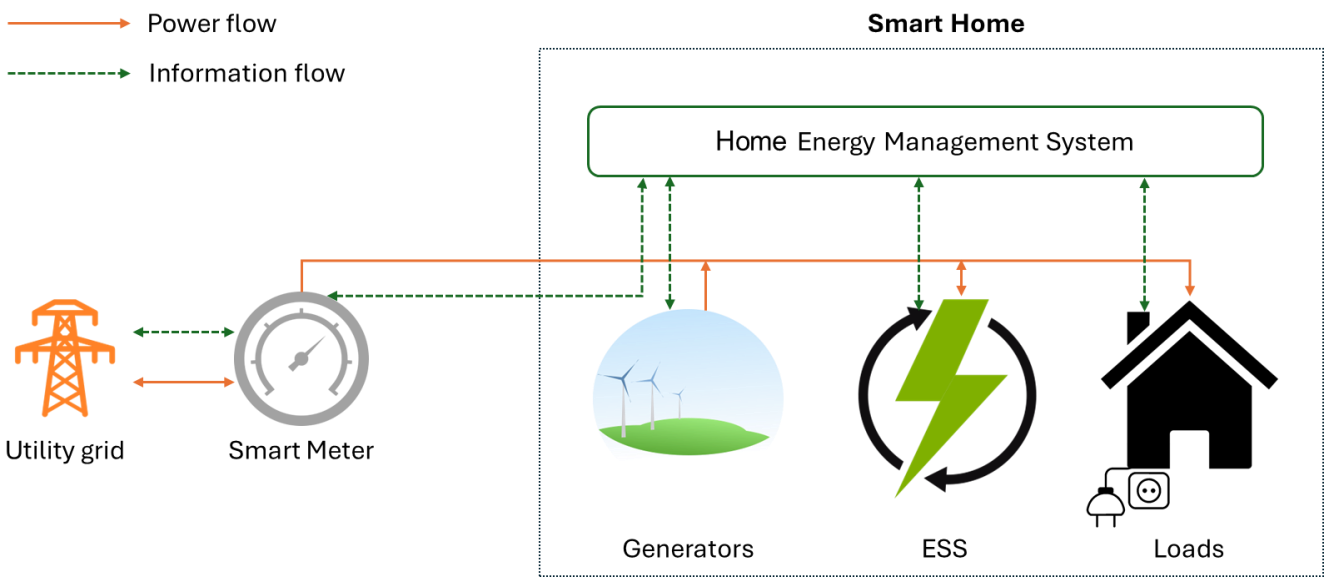

Figure 1. Illustration of smart home

The system is composed of the following key elements:
Utility Grid: This represents the external, main power grid from which the home can purchase electricity. In our model, the grid also allows for the selling of surplus energy (e.g., from solar panels or the battery), creating an economic incentive for intelligent energy management.
Smart Meter: Positioned at the interface between the home and the utility grid, the smart meter measures the bidirectional flow of electricity. It provides critical information about the amount of energy being bought from or sold to the grid, which is used to calculate the total energy cost.
The Smart Home (Dashed Boundary): This encapsulates all the components managed by our system.
Home Energy Management System (HEMS): This is the central controller, the "brain" of the entire system. In this project, the HEMS is implemented as our Deep Reinforcement Learning (DRL) agent. It continuously receives information from all other components, processes this information (the "state"), and makes optimal decisions (the "actions") to control the flow of energy within the home.
On-site Renewable Generation: This component represents distributed energy resources like rooftop solar panels or residential wind turbines. It provides a source of free, albeit intermittent and unpredictable, energy.
Energy Storage System (ESS): This is a rechargeable battery that plays a crucial role in the optimization strategy. It can be charged using surplus renewable energy or cheap grid electricity and discharged later to power the home when grid electricity is expensive.
Loads: This represents all electricity-consuming appliances in the home, the primary controllable load of this project: the HVAC system.
The interactions within the system are defined by two types of flows:
Power Flow (Orange Solid Lines): This represents the physical flow of electrical energy. Power can flow from the grid and the renewable sources to the loads and the ESS. Conversely, power can flow from the ESS to the loads, and any excess power from renewables or the ESS can be sold back to the grid.
Information Flow (Green Dashed Lines): This represents the communication and control signals. The HEMS gathers real-time data—such as the price of electricity, the current state-of-charge of the ESS, the output of the renewable generator, and the indoor temperature—to form its understanding of the current

state. Based on this state, the HEMS sends control commands to the ESS (instructing it to charge or discharge) and to the HVAC system (setting its power level).

1.4 Workflow:

The project will be executed following these steps:

Figure 2. Workflow ot this project

2. DATASET AND PREPROCESSING

2.1 Dataset:

For this project, I used the well-known Pecan Street Inc. database, which provides high-resolution, real-world energy data from residential homes. I focused on data from a single home in Austin, Texas, for the summer period of 2018, when HVAC usage is highest. The specific data streams I extracted were:
Solar Generation (kW): Represents renewable energy supply (pt).
Non-shiftable Load (kW): Represents the base electricity demand from uncontrollable appliances (bt).
Outdoor Temperature (°F): Represents an external environmental factor (T_out).
Electricity Price ($/kWh): Represents the dynamic cost from the utility (vt).
To ensure my model could generalize to new situations, I split the data chronologically: data from June and July 2018 was used for the training phase, and data from August 2018 was reserved as an unseen test set for the final evaluation.

Raw dataset head:
Table 1. Table of raw dataset

Unnamed	Timestamp	Energy_ Generation	Energy_ Consumption	SMP	EV_ Consimption	Time_of_day
0	2018-10-03 21:00:00	0.0	0.8	0.06814	0.0	0.875000
1	2018-10-03 21:15:00	0.0	0.7	0.06814	0.0	0.885417
2	2018-10-03 21:30:00	0.0	0.7	0.06814	0.0	0.895833
3	2018-10-03 21:45:00	0.0	0.7	0.06814	0.0	0.906250
4	2018-10-03 22:00:00	0.0	0.6	0.06812	0.0	0.916667

2.2 Preprocessing:

Neural networks train more effectively when their input data is on a consistent scale. Therefore, I implemented a preprocessing step to normalize the state features. Before being fed into the agent's networks, each component of the state vector was scaled to a numerical range of **[0, 1]** using min-max normalization. The scaling parameters (min and max values) were calculated from the training data and then applied to both the training and testing data to prevent data leakage.

2.3 State Representation:

The agent's ability to make good decisions depends entirely on the information it receives. I designed the state representation to provide a comprehensive snapshot of the environment at each time step ***t***. The state, ***st***, is a 7-dimensional vector:

st = (pt, bt, Bt, Tout, Tt, vt, t')

pt (Solar generation) and ***bt*** (Non-shiftable load): Inform the agent about the immediate power supply and demand.
Bt (ESS energy level): A critical internal state indicating available stored energy.
Tout (Outdoor temperature) and Tt (Indoor temperature): Essential for making thermal comfort decisions.
vt (Electricity price): The primary driver for cost-saving decisions.
t' (Time of day, 0-23): Allows the agent to learn daily cyclical patterns in price and temperature.

3. MODEL

3.1 Parameters:

To create a realistic simulation environment, I configured the system with the following physical and system parameters, based on the reference paper:

Table 2. Table of parameters

Energy Storage System (ESS)	Max Capacity (***Bmax***): 6 kWh Min Capacity (***Bmin***): 0.6 kWh Max Charge/Discharge Rate (***cmax/dmax***): 3 kW Efficiency (***ηc/ηd***): 95%
HVAC System	Max Power (***emax***): 2 kW
Thermal Comfort Zone:	Min Temperature (***Tmin***): 19°C Max Temperature (***Tmax***): 24°C
Building Thermal Dynamics Model:	Thermal Inertia (***ε***): 0.7 Heat Transfer Coefficient (***A***): 0.14 kW/°F

3.2 DDPG Agent Model Description:

3.2.1 Architecture

My DDPG agent is composed of two primary neural networks, the Actor and the Critic, each with a corresponding target network for stable learning.

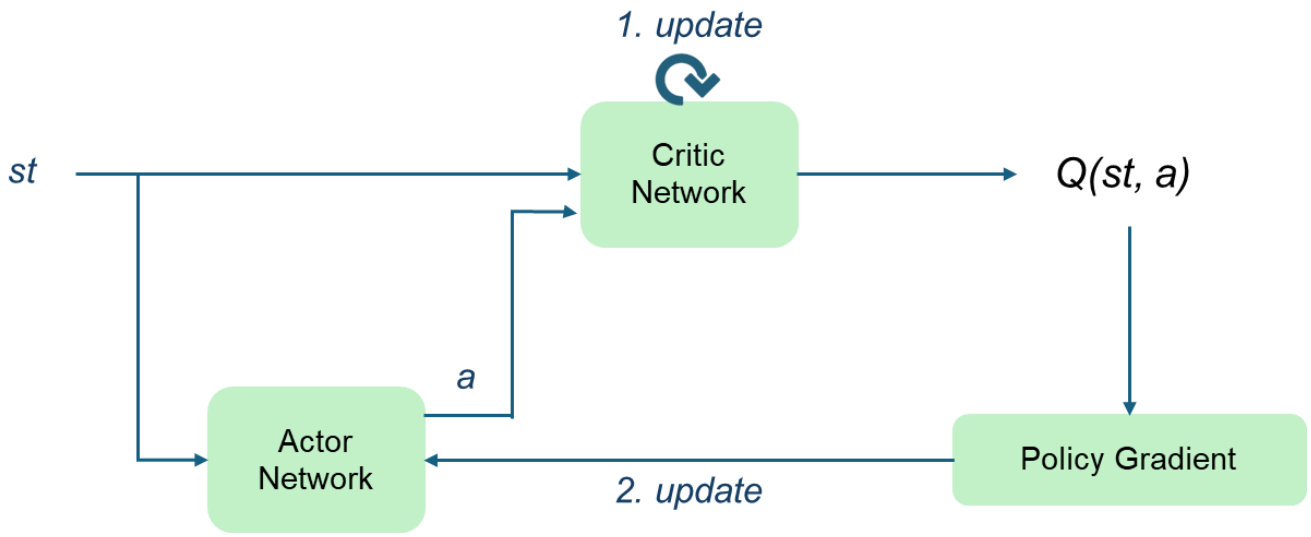

Figure 3. Actor and Critic

Actor Network:

- Input: The 7-dimensional state vector st.
- Architecture: Two fully-connected hidden layers with 300 and 600 neurons, respectively, using the ReLU activation function.
- Output: A 2-dimensional continuous action at = (ft, et). I used a tanh activation for the ESS action ft because its [-1, 1] range is ideal for representing both charging (positive) and discharging (negative). For the HVAC action et, I used a sigmoid activation, as its [0, 1] range naturally maps to a power level that is always non-negative. These outputs are then scaled to their real-world physical limits.

Critic Network:

- Input: Both the state vector st and the action vector at proposed by the Actor.
- Architecture: Four hidden layers (300, 600, 600, 600 neurons). The state is processed first, and the action vector is then concatenated and processed through the subsequent layers.
- Output: A single scalar value, Q(s, a), which estimates the expected long-term cumulative reward of taking that action in that state.

3.2.2 Training Algorithm

I implemented the DDPG training algorithm, which centers around the interaction between the four networks and a Replay Memory buffer. The process I followed for training is:

- The Actor network determines an action based on the current state. To encourage the discovery of new strategies, I added exploration noise to this action.
- This action is executed in my simulated environment, which returns the next state and a reward.
- This entire experience tuple (state, action, reward, next_state) is stored in the Replay Memory. Storing experiences and sampling from them later helps to break the correlation between consecutive data points, stabilizing the learning process.
- For the update step, a mini-batch of experiences is randomly sampled from the memory.
- The Critic network is updated by minimizing the difference between its Q-value prediction and a more stable "target" Q-value calculated using the target networks.

- The Actor network is then updated by following the gradient provided by the Critic, effectively pushing the Actor to produce actions that the Critic evaluates more highly.
- Finally, the weights of the target networks are slowly and smoothly updated to track the main networks, which is a key technique for maintaining stability.

3.2.3 Hyperparameters

The following hyperparameters were used to configure the training process:

Table 3. Table of hyperparameters

Learning rates	Actor: a_actor = 0.0001 Critic: a_critic = 0.001
Discount factor	Gamma = 0.995
Replay Memory Size	2000
Mini-batch size	32
Target network soft update rate	0.001
Number of training episodes	200 – 2000

3.2.4 Rewards after training

Here are the figures of episode reward after 200 and 2000 episodes.

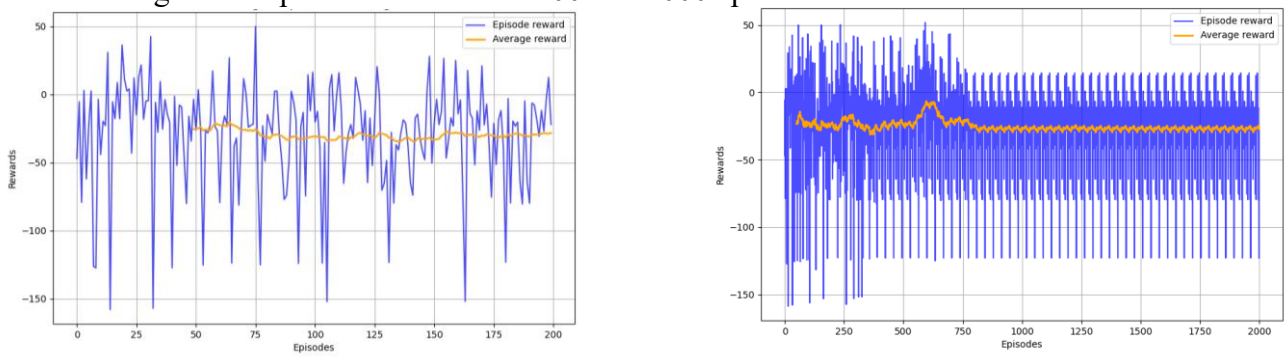

Figure 4. Reward after 200 and 2000 episodes

Analyzing the two reward plots, a clear progression in the agent's learning and performance is evident as the training duration increases. The plot for 200 training episodes indicates an early stage of learning, characterized by significant volatility in the "Episode reward" (blue line). The "Average reward" (orange line), while showing an initial upward movement, remains noisy and does not settle into a clear, stable trend, suggesting that the agent's policy is still highly inconsistent and far from optimal. In stark contrast, the plot for 2000 training episodes demonstrates substantial improvement. The "Average reward" line (orange) is considerably smoother and exhibits a much more defined upward trajectory, largely stabilizing at a higher (less negative) reward value. This indicates that with extended training, the agent has learned a significantly more effective and consistent policy, leading to a much better performance outcome and a clear sign of convergence compared to the shorter training run.

4. EXPERIMENT SETUP PROPOSED

4.1 Baseline 1: ON/OF Controller

To establish a performance benchmark, I first implemented a simple, rule-based ON/OFF controller. This controller does not use the ESS. It turns the HVAC on at full power if the indoor temperature exceeds Tmax and switches it off completely if the temperature falls below Tmin. This baseline represents a common, non-intelligent control strategy found in many traditional thermostats.

4.2 Baseline 2: DDPG HVAC – Only Controller (without ESS)

To isolate and measure the specific value added by the ESS, I implemented a second baseline. This model uses the exact same DDPG architecture and training procedure as my proposed model, but with the ESS functionality disabled (cmax and dmax set to 0). By comparing the performance of the proposed model against this baseline, I can directly quantify the cost savings attributable to intelligent battery management.

5. CONCLUSION

5.1 Results

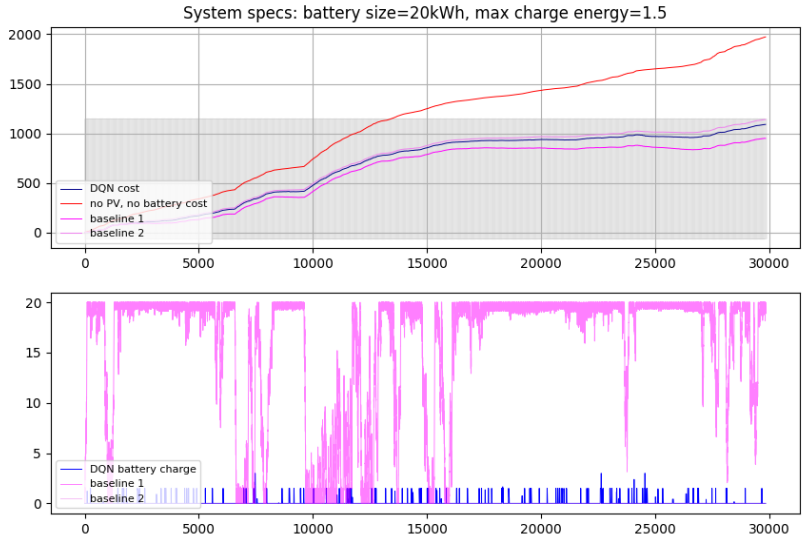

Figure 5. Result of this system

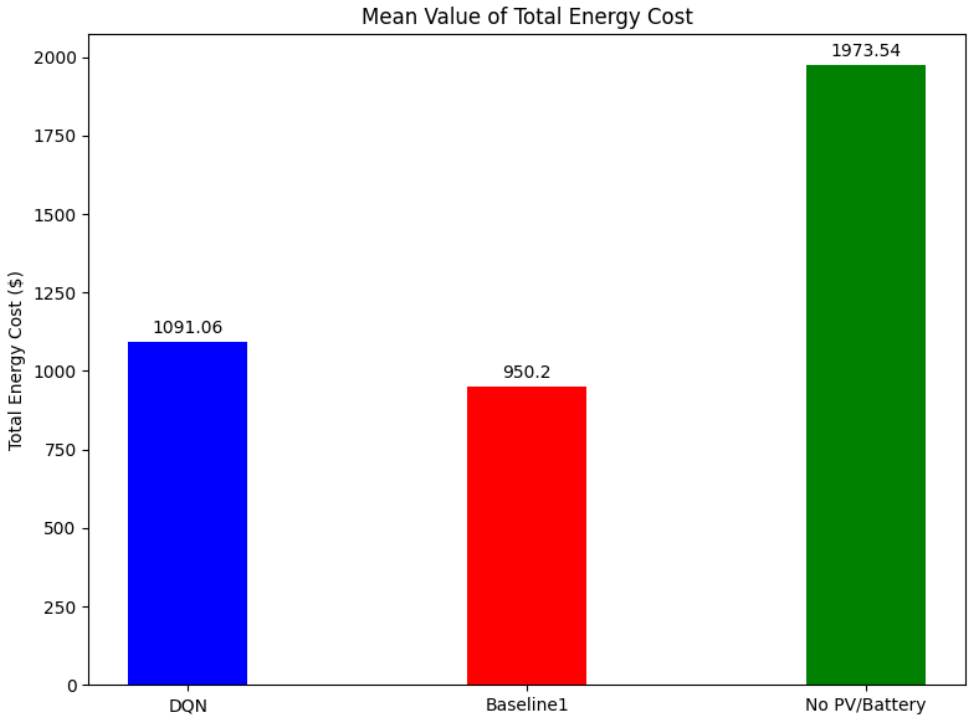

Figure 6. Mean value of total energy

5.2 Conclusion

To isolate and measure the specific value added by the ESS, I implemented a second baseline. This model uses the exact same DDPG architecture and training procedure as my proposed model, but with the ESS functionality disabled (cmax and dmax set to 0). By comparing the performance of the proposed model against this baseline, I can directly quantify the cost savings attributable to intelligent battery management.

Two distinct lines are presented in the plot:

Episode Reward (Blue Line): This line shows the individual reward achieved in each specific training episode. As is typical in reinforcement learning training, particularly during initial phases or when dealing with environmental variability, this line exhibits considerable fluctuations and noise. Despite this variability, a general upward trend can be observed, suggesting that the agent is progressively learning to accumulate higher rewards over time.

Average Reward (Orange Line): This line represents a smoothed average of the episode rewards, likely calculated as a rolling mean over a defined window of episodes (e.g., 50 episodes as often implemented in such systems). The "Average reward" curve provides a clearer, more stable representation of the overall learning progression. It demonstrates a consistent and significant increase in the average reward as the training episodes advance.

REFERENCE:

[1] https://arxiv.org/pdf/1909.10165v2
[2] D. Zhang, S. Li, M. Sun, and Z. O'Neill, "An optimal and learning-based demand response and home energy management system," IEEE Trans. on Smart Grid, vol. 7, no. 4, pp. 1790-1801, July 2016
[3] [1901.04693] Energy-Efficient Thermal Comfort Control in Smart Buildings via Deep Reinforcement Learning