DQN-Based Coordinated HVAC and Battery Control for Cost-Optimal Residential Heating

Carmen Burunciuc

***Abstract*—The research presents a Deep Q-Network (DQN) system which governs the coordinated operation of home HVAC systems and battery systems to meet changing electricity costs. The problem uses a Markov decision process that employs a brief state vector which contains indoor temperature outdoor temperature solar input price state-of-charge and time-of-day information and uses a discrete joint action space that combines HVAC power with battery charging and discharging operations. The proposed reward system evaluates energy expenses together with comfort violations and battery usage. The simulation experiments showed that running costs decreased when compared to a rule-based system while maintaining excellent comfort standards.**

I. Introduction

RESIDENTIAL users and small structures present considerable opportunities for energy flexibility, particularly when utilizing electric heating systems (heat pumps), onsite generation (photovoltaic), and storage solutions (batteries). In a setting of fluctuating energy prices, the aim of an energy management system is twofold: to ensure thermal comfort remains within acceptable ranges and to lower operating expenses via control choices made at brief intervals. In reality, the environment is characterized by uncertainty (temperature, solar radiation, energy costs), and achieving comprehensive and calibrated modeling for each residence is challenging to do reliably. In these circumstances, traditional optimization techniques may prove challenging to implement in real-time, particularly when the model and constraints are intricate or vary with time.

Deep reinforcement learning (DRL) has emerged as an appealing option for sequential control challenges in energy systems since it can develop control strategies directly through interactions with the environment without needing a fully specified model. Nonetheless, DRL frequently encounters low exploration efficiency and extended training durations, particularly when there are constraints (comfort, equipment usage) and expansive action spaces. A crucial approach to speeding up learning is blending domain knowledge (such as heuristic rules) into the exploration and/or sampling method from the experience memory, allowing the agent to gather more ”valuable” transitions during the initial training stages. .

The source material I utilize suggests a ”knowledge penetration” approach in which a rule-based policy (Rule-Based Controller, RBC) aids in action generation during training, with the balance among random exploration, rule-guided exploration, and the exploitation of the acquired policy managed by exponential probability functions. The process of experience replay

C. Burunciuc is with Faculty of Automatic and Computer Science, Technical University of Cluj-Napoca, Cluj-Napoca, Romania, e-mail: Burunciuc.Ma.Carmen@student.utcluj.ro.

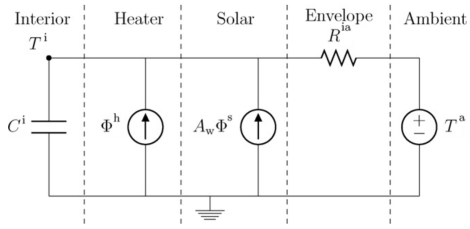

Fig. 1. RC model

sampling has been modified to select recent transitions over older ones because this change aims to improve the speed of learning. [1]

The DRL-based energy management framework which I developed through this paper uses a public repository which simulates building thermal performance through an RC-type thermal behavior model 1 and shows building thermal performance across three systems which include a heat pump system and a photovoltaic system and a market price system. The primary contribution to the fundamental implementation is the clear integration of the battery charge/discharge cycle by incorporating the state of charge (SOC) into the environmental state and factoring in the energy cost tied to the energy exchange with the battery within the reward function. My implementation uses the agent's action to simultaneously control HVAC systems and manage battery operations which include charging and discharging while the system updates state of charge at each time step based on operational limits and efficiency performance..

The objective is to develop an agent that can reduce the total energy expenses which include charging and discharging costs while maintaining indoor temperatures at comfortable levels during outdoor temperature changes and solar energy production and energy price fluctuations. The document presents a repeatable system which operates HVAC systems together with battery management systems while testing how energy storage solutions impact learning performance and operational costs with the experimental section showing training data and evaluation results.

II. Related Work

Zhang et al. (2023) [1] introduce a knowledge-infused DRL approach for the online optimization of integrated demand response within a multi-energy (electricity–gas) residential energy framework, where traditional DQN is enhanced by incorporating domain expertise through a rule-based controller (RBC). The core concept is that the agent must not depend

only on random exploration; instead, it should produce "superior" transitions through actions suggested by rules. The equilibrium among random exploration, rule-based exploration, and the exploitation of the established policy should be managed using exponential probability functions. Furthermore, the authors adjust the sampling from experience replay so that recent transitions are chosen with greater likelihood (exponential distribution) to enhance training efficiency. In tests conducted in a standard multi-energy household setting, the approach decreases training duration by approximately 48.78% relative to conventional DQN and results in reduced energy expenses compared to the uncontrolled situation and an integrated RBC (showing 26.17% and 9.88% savings)

III. Methodology

A. Project Starting Point and Evolution

Our implementation utilizes the public repository heating-rl-agent (Cernewein, 2024) [2], accessible on GitHub, which offers a thorough and reproducible framework for optimal energy management of a residential building leveraging deep reinforcement learning algorithms (DQN and DDPG). The initial system simulates the thermal dynamics of a structure using a sophisticated analogy with RC (Resistance-Capacitance) electrical circuits, incorporates a controllable heat pump, a solar panel for onsite energy production, and access to fluctuating spot energy prices, aiming to reduce heating operational costs while ensuring the thermal comfort of residents.

Nonetheless, while the foundational code conceptually refers to the inclusion of an energy storage system (battery) as a possible element, the initial execution did not comprehensively and methodically incorporate the battery within the dynamics of the simulation environment, in the state space perceived by the agent, into the reward function, or within the techniques for adjusting internal models. Battery energy was the sole passive variable considered, without any direct depiction of state of charge (SOC), real conversion efficiencies, or operational limitations.

The primary contribution of my study includes:

1) Broadening the state space to incorporate the battery's state of charge (SOC) as an ongoing and pertinent observation variable, enabling the agent to make well-informed choices regarding the storage strategy.
2) Reconfiguring the action space with a composite coding that enables simultaneous and coordinated management of both the heat pump (HVAC) and the battery charging/discharging unit, rather than addressing them as separate issues;
3) Adjusting the reward function to explicitly penalize net energy expenses (now considering inherent battery conversion losses), distinguishing between charging and discharging, and incorporating actual system efficiencies;
4) Incorporating authentic battery dynamics, featuring SOC update formulas that consider conversion losses, operational boundaries (a battery should not be discharged below 10% for durability or charged above 90%), and the principles of energy conservation.

These modifications shift the issue from a passive battery HVAC control issue to an HVAC–Battery co-optimization challenge where the agent has to develop a control strategy that reconciles heating, storage, and energy use according to weather conditions and pricing indications, [3].

B. Detailed Description of the Simulation Environment

1) Building Thermal Model: RC (Resistance-Capacitance) Analogy: Any structure can be represented through an analogy with an RC electrical circuit. The fundamental concept is that:

- **Thermal resistance (R)** is the property of materials to oppose heat flow (similar to how electrical resistance opposes current flow);
- **Heat capacity (C)** is the amount of energy required to change the temperature of a material by 1 Kelvin (analogous to an electrical capacitor).

For our building, we use a simplified model with two thermal masses:

- **Interior Temperature (interior air)**: temperature T_i^t, capacity $C_i = 10$ kWh/°C
- **Tire mass (walls, windows)**: temperature T_{env}^t, capacity C_{env} (implicit)

Heat transfer through building elements is governed by the laws of physics:

$$\frac{dT_i}{dt} = \frac{1}{C_i}\left[\Phi^h + A_w\Phi^s + \frac{T_{\text{env}} - T_i}{R_{ia}}\right] \tag{1}$$

unde:

- Φ^h — thermal power supplied by the heat pump (kW);
- A_w — effective area of windows for solar gain (m^2);
- Φ^s — density of incident solar flux per m^2 (kW/m^2);
- T_{env} — tire temperature (intermediate);
- $R_{ia} = 0.05$ K/kW — thermal resistance between the interior and the tire.

For discrete implementation (with time steps of 1 hour = 3600 seconds), the equation becomes:

$$T_i^{t+1} = T_i^t + \frac{\Delta t}{C_i \cdot R_{ia}}\left[\Phi^h(t) + A_w\Phi^s(t) - (T_i^t - T_a^t)\cdot\frac{1}{R_{ia}}\right] \tag{2}$$

Term $(T_i^t - T_a^t)/R_{ia}$ represents the heat flow lost through the envelope to the outside. This is a critical component of the model: as the difference between the indoor and outdoor temperatures increases, the building loses more heat, forcing the heat pump to work harder.

2) Heat Pump and Energy Conversion: A heat pump is a device that extracts heat from the outside environment (air, ground, water) and transfers it indoors. The efficiency of this process is characterized by **Coefficient of Performance (COP)**, define as:

$$\text{COP} = \frac{\text{Thermal power supplied}}{\text{Electric power consumed}} \tag{3}$$

For an air-to-air heat pump in a temperate climate, the COP is typically between 2.5 and 4.0. In our simulation, we use an approximately constant COP. COP ≈ 3.0, which means:

$$\Phi^h = \text{COP} \times P_{\text{in}} = 3.0 \times P_{\text{in}} \tag{4}$$

The electrical input power is controlled by the agent and can vary between 0 and $P_{\max} = 1.6$ kW, through discreet action $a_{\text{hvac}} \in \{0, 0.5, 1.0\}$:

$$P_{\text{in}}(t) = a_{\text{hvac}}(t) \times P_{\max} = a_{\text{hvac}}(t) \times 1.6 \text{ kW} \tag{5}$$

It offers three levels of operation:

- **OFF** ($a_{\text{hvac}} = 0$): $P_{\text{in}} = 0$ kW, $\Phi^h = 0$ kW;
- **HALF** ($a_{\text{hvac}} = 0.5$): $P_{\text{in}} = 0.8$ kW, $\Phi^h = 2.4$ kW;
- **FULL** ($a_{\text{hvac}} = 1.0$): $P_{\text{in}} = 1.6$ kW, $\Phi^h = 4.8$ kW.

This discreet design (with 3 levels) is motivated by:

1) Operational simplicity: actual equipment often operates in predefined modes;
2) Stability: reduces the complexity of the action space and improves training convergence;
3) Energy reality: modern heat pumps are also compressor-based with discrete states.

3) Solar Radiation and Passive Capture: Solar radiation entering through windows contributes to interior heating. The captured solar power is:

$$\Phi^s(t) = A_w \times \alpha_w \times G^s(t) \tag{6}$$

where:

- A_w — total window area (m^2);
- α_w — the absorption-transmission coefficient of windows (typically 0.6–0.8);
- $G^s(t)$ — global horizontal solar irradiance (W/m^2), variable by time and day.

In the simulation, irradiance is provided directly from meteorological data in **Renewables.Ninja**, and is already integrated into a 1-hour window.

4) Observation Space: At each step t, the agent observes a state composed of **6 continues variables**:

$$s_t = [T_i^t, T_a^t, \Phi_t^s, p_t, \text{SOC}_t, \text{TOD}_t] \tag{7}$$

Details of each component:

- T_i^t (Indoor temperature, °C): Typical values 15–25°C. This variable is central to comfort assessment; the agent directly observes the physical state of the building.
- T_a^t ((Outside temperature, °C): Typical values –10–30°C (depending on the season). This determines heat loss and the need for heating.
- Φ_t^s (Solar power, kW): Typical values 0–2.5 kW for a small panel. This provides information about the availability of passive and renewable energy sources.
- p_t (Spot price of energy, €/MWh or RON/MWh): Typical values 15-55 €/MWh. This signals to the agent when it is "cheap" to consume energy (for heating and battery charging) and when it is "expensive" (forcing savings)..
- SOC_t (Battery charge status, dimensionless [0–1]): Typical values 0.1–0.9. This is **new variable** added by us. It allows the agent to know how "full" the battery is and make informed decisions about charging vs. discharging.
- TOD_t (Time of day, cyclic encoding): Usually encoded as $[\sin(2\pi t/24), \cos(2\pi t/24)]$ to capture the periodic nature of the day. The agent can learn patterns such as "what happens at 8 a.m. vs. 8 p.m."

Why are these observations important? The agent must learn complex correlations. For example:

- If T_i is low and p_t is low, charge the battery and use HVAC;
- If T_i is low and p_t is high, discharge the battery (if it has SOC) and minimize HVAC;
- If Φ_t^s is high and SOC_t isn low, charge the battery from surplus solar energy.

C. Battery Integration: The Detailed Energy Storage Model

1) Physical Parameters of the Battery: The modeled storage system has the following nominal parameters:

$$E_{\text{bat}} = 10 \text{ kWh} \tag{8}$$

This is a typical capacity for a house with a modest solar installation. The operational limits are:

- $\textbf{SOC}_{\min}$ = 0.1 (avoid deep discharge, which degrades Li-ion batteries);
- $\textbf{SOC}_{\max}$ = 0.9 (avoid overloading).

2) SOC Update Equations and Energy Losses: The battery command in one action is $a_{\text{bat}} \in \{-3, -2, -1, 0, 1, 2, 3\}$ kW, where:

- Values **negative = discharge** (the battery supplies energy to the grid/consumer);
- Values **pozitive = charge** (the network/sources supply energy to the battery);
- **0** = inactivity.

For each time step, the energy required from the battery is:

$$e_{\text{bat}}^{\text{req}}(t) = p_{\text{bat}}(t) \times \Delta t \tag{9}$$

where $\Delta t = 1$ hour = 3600 seconds. For example, if $p_{\text{bat}} = 2$ kW for an hour, energy is $e_{\text{bat}}^{\text{req}} = 2$ kWh.

Due to **intrinsic DC-AC conversion losses and magnetic friction**, the energy actually stored or extracted is lower:

$$e_{\text{bat}}^{\text{stored}}(t) = \begin{cases} \dfrac{e_{\text{bat}}^{\text{req}}(t)}{\eta_{\text{ch}}} & \text{if } p_{\text{bat}}(t) > 0 \text{ (charging)} \\ e_{\text{bat}}^{\text{req}}(t) \times \eta_{\text{disch}} & \text{if } p_{\text{bat}}(t) < 0 \text{ (discharging)} \end{cases} \tag{10}$$

The rates used are:

- $\eta_{\text{ch}} = 0.90$ — charging efficiency (10% losses));
- $\eta_{\text{disch}} = 0.95$ — discharge efficiency (5% losses).

This reflects the fact that **charging is less efficient** (requires more energy from the grid than is actually stored), while discharging is slightly more efficient.

Numerical example:

- If the agent orders $p_{\text{bat}} = 2$ kW one hour charge:
  - Energy required: $e_{\text{req}} = 2$ kWh;
  - Stored energy: $e_{\text{stored}} = 2/0.90 = 2.22$ kWh;

- **From the grid**: 2 kWh; **Only** $2/0.90 = 2.22$ kWh enters the battery (so the grid actually supplies 2.22 kWh);

- If the agent orders $p_{\text{bat}} = -2$ kW discharge:
  - Energy required: $e_{\text{req}} = -2$ kWh;
  - Energy supplied by the battery: $e_{\text{stored}} = -2 \times 0.95 = -1.9$ kWh;
  - **From the grid**: 1.9 kWh; **The battery loses 0.1 kWh** as heat/losses..

The charging status is updated according to:

$$\text{SOC}^{t+1} = \text{SOC}^{t} - \frac{e_{\text{bat}}^{\text{stored}}(t)}{E_{\text{bat}}} \tag{11}$$

Then, the values are **clipped** within the valid range to ensure safe operation:

$$\text{SOC}^{t+1} = \text{clip}(\text{SOC}^{t+1}, 0.1, 0.9) \tag{12}$$

If the agent attempts to command a charge that would exceed $\text{SOC}_{\max}$, the excess is simply ignored (the battery no longer accepts energy). Similarly, if it commands a discharge below $\text{SOC}_{\min}$, the discharge is partially or completely canceled.

3) Composite Actions: HVAC + Battery: The action space is redefined as a **two-dimensional discrete composite action**:

$$a_t = (a_{\text{hvac}}, a_{\text{bat}}) \in \mathcal{A} \tag{13}$$

where:

- $a_{\text{hvac}} \in \{0, 0.5, 1.0\}$ — 3 levels of HVAC power (OFF, HALF, FULL);
- $a_{\text{bat}} \in \{-3, -2, -1, 0, 1, 2, 3\}$ — 7 battery power levels (kW).

The Cartesian product gives:

$$|\mathcal{A}| = 3 \times 7 = 21 \text{ discrete actions} \tag{14}$$

This dimension is **deliberately kept small** in order to:

1) Accelerate training convergence (DQN with 21 outputs vs. 100+ is much more stable);
2) Reflect reality: commercial equipment has discrete modes;
3) Facilitates interpretability: easy to analyze what strategy the agent has learned.

D. Reward Function

Reward at every time step integrates **three competing components** which shape conflicting goals:

$$r_t = -\lambda_P \cdot c_t - \lambda_C \cdot \text{pen}_{\text{comfort}} - \lambda_B \cdot \text{pen}_{\text{bat}} \tag{15}$$

where λ_P, λ_C, λ_B are **relative weights** (hyperparameters).

1) Energy Cost Component (c_t): The net energy cost integrates heat pump consumption and battery exchange:

$$c_t = [P_{\text{hvac}}(t) + P_{\text{net}}^{\text{bat}}(t)] \times p_t \times \Delta t \tag{16}$$

where:

- $P_{\text{hvac}}(t)$ — electric power consumed by the heat pump = $a_{\text{hvac}}(t) \times P_{\max}$ (kW);
- $P_{\text{net}}^{\text{bat}}(t)$ — **net** power which exchanges energy with the battery. This is different from the command a_{bat}:
  - If $a_{\text{bat}} > 0$ (charge), $P_{\text{net}}^{\text{bat}} = a_{\text{bat}}/\eta_{\text{ch}}$ (the grid provides more);
  - If $a_{\text{bat}} < 0$ (discharge), $P_{\text{net}}^{\text{bat}} = a_{\text{bat}} \times \eta_{\text{disch}}$ (the grid receives less due to losses).
- p_t — spot price in €/MWh;
- Δt — time interval (1 hour).

Physical intuition:

- Negative cost (negative reward = penalty) encourages the agent to consume less gross energy;
- The agent learns to **shift loading** to low-cost hours and **unloading** to high-cost hours;
- Battery efficiency makes discharging more "profitable" than charging (less loss).

2) Penalty for Exceeding Comfort Levels ($\text{pen}_{comfort}$): Thermal comfort is defined as maintaining the indoor temperature within the range **[19°C, 23°C]**. Any deviation from this range incurs a proportional penalty:

$$\text{pen}_{\text{comfort}} = \begin{cases} \max(0, 19 - T_i^t) + \max(0, T_i^t - 23) & \text{if } T_a^t > T_{\text{threshold}} \\ 0 & \text{else} \end{cases} \tag{17}$$

Explication:

- $\max(0, 19 - T_i^t)$ — if the temperature falls below 19°C, the penalty is proportional to how many degrees below 19;
- $\max(0, T_i^t - 23)$ — if the temperature rises above 23°C, a penalty proportional to the excess;
- Condition $T_a^t > T_{\text{threshold}}$ Avoid strange penalties: if it's very cold outside and we don't have heating, I don't penalize the agent for not being able to maintain 19°C.

The weight λ_C is chosen high (usually 1000+) to ensure that the agent prioritizes comfort over savings.

3) Penalty for Battery Activation (pen_{bat}) — Optional: To discourage "excessive cycling" that would degrade the battery:

$$\text{pen}_{\text{bat}} = |p_{\text{bat}}(t)| \tag{18}$$

This penalizes any non-zero charging or discharging, encouraging the agent to leave the battery alone when it is not useful. The weight λ_B is usually small (0.01–0.1) so as not to inhibit the usefulness of the battery.

4) Final Normalization: The gross reward can vary widely; to stabilize training, I normalize it:

$$r_t^{\text{final}} = \text{clip}(r_t, -1, 1) \tag{19}$$

This ensures that rewards remain within $[-1, 1]$, avoiding gradient explosions in the neural network.

E. Input Data and External Sources

1) Meteorological data: Renewables.Ninja: Outdoor temperatures and solar radiation come from the database **open-source Renewables.Ninja** [4], which provides hourly weather data for any geographic location.

In our simulation:

- Location: **Denmark** (55.68°N, 12.54°E) — a country with temperate climates and marked seasonal variability;
- Year: **2014** — full year with winter and summer seasons;
- Rezolution **Hourly** (8760 data points per year).

From the complete annual set, at **start of each training episode**, **I randomly select a window of 24 consecutive hours**:

$$\text{weatherstart} = \text{random}(1, 365 - 24) \times 24 \text{ [hours]} \tag{20}$$

This ensures:

1) **Training variability:** The agent does not "memorize" a specific day, but learns generalizable strategies;
2) **Representativeness:** May encounter cold winter season, mild autumn, hot summer, etc.;
3) **Reproducibility:** With seeded randomness, experiments are repeatable.

2) Spot Energy Prices: NordPool and OPCOM: Electricity prices are fundamental to the battery arbitrage strategy. They come from real markets:

- **NordPool** (for international calibration): DK1/DK2 region in Denmark, 2014;
- **OPCOM** (optional for local relevance): the electricity market in Romania.

Typical characteristics of spot prices:

- Varies **hourly**: ex. €15/MWh at night (low demand), €45/MWh at 8 a.m. (peak);
- Pattern **clear diurnal**: low price at night, rising in the morning, peaking at noon/evening;
- Pattern **seasonal**: winter is more expensive (more heating), summer is cheaper (more solar energy).

Similar to weather data, when initializing the episode, I randomly select a 24-hour window from the available year:

$$\text{pricestart} = \text{random}(0, \text{len(prices)} - 24) \tag{21}$$

Example of use:

- If the agent observes that the price = €20/MWh at 2 AM, it can charge the battery now;
- At 8 AM when the price = 50 €/MWh and T_i is low, it can discharge the battery to avoid purchasing expensive energy;
- The difference of €30/MWh per 2 kWh discharged = €60 saved, with efficiency cost = losses of ∼5–10% (0.1–0.2 kWh).

3) Local Consumption and Production Tasks: **Basic Consumption (Non-Controllable Load)**

A house always has minimal consumption for appliances, lighting, etc., regardless of the agent's actions:

$$P_{\text{base}}(t) \approx 0.2 - 0.5 \text{ kW (diurn) la } 0.05 - 0.1 \text{ kW (nocturn)} \tag{22}$$

This is not controllable and must be paid for from the grid or battery.

Photovoltaic Production

Solar panel with nominal installed power ∼3 kW:

$$P_{\text{PV}}(t) = P_{\text{nom}} \times \frac{G^s(t)}{G^{\text{std}}} \tag{23}$$

where $G^{\text{std}} = 1000$ W/m^2 is the standard irradiance. The output is:

- **Night:** 0 kW;
- **Cloudy day:** 0.2–0.8 kW;
- **Sunny day:** 1.5–3 kW.

Controllable Power: Heat Pump and Battery

- **Heat Pump:** 0–1.6 kW (3 discrete levels);
- **Batery:** –3 to +3 kW (7 discrete levels).

Daily energy balance:

Every hour, the grid must provide:

$$P_{\text{grid}} = P_{\text{HVAC}} + P_{\text{base}} + P_{\text{net}}^{\text{bat}} - P_{\text{PV}} \tag{24}$$

If $P_{\text{grid}} > 0$, consume from the grid (positive cost). If $P_{\text{grid}} < 0$, I export the solar surplus (negative income, i.e., cost reduction).

F. Learning Algorithm: Deep Q-Network (DQN)

1) Fundamental Principles of DQN: DQN is a value-based reinforcement learning algorithm that learns an approximation of the Q-value function:

$$Q(s, a) \approx \mathbb{E}[R_t | S_t = s, A_t = a] \tag{25}$$

where $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the return (discounted cumulative reward) if I take action a in state s.

The idea is that if I know the exact Q-value for each pair (state, action), then **the optimal policy is trivial.**:

$$\pi^*(s) = \arg\max_a Q^*(s, a) \tag{26}$$

That is, in each state, I take the action with the highest estimated Q value.

2) Neural Network Architecture: The Q function is approximated by a fully-connected neural network (MLP).:

Input Layer: 6 neurons (state s_t)
↓ [6]
Dense(128, ReLU)
↓ [128]
Dense(128, ReLU)
↓ [128]
Output Layer: 21 neurons (Q-values for 21 actions)
↓ [21]

Motivation for architecture:

- **6 inputs:** State dimension ($T_i, T_a, \Phi_s, p, \text{SOC}, \text{TOD}$);

- **2 hidden layers of 128 neurons:** Complex enough to approve nonlinear relationships in data; not too deep to avoid vanishing gradients;
- **ReLU activation:** $\max(0, x)$ — Linear rectified, easy to drive, and avoids saturation of gradients;
- **21 outputs:** A Q-value for each discrete action in space.

3) The Experience Replay Mechanism: DQN uses **experience replay buffer** to decouple learning data collection:

1) **Collection:** As the agent interacts with the environment, it stores **tranzitions** $(s_t, a_t, r_t, s_{t+1}, \text{done})$ in a circular buffer with fixed capacity (e.g., 10,000 transitions);
2) **Sampling:** At every step of training, I extract a **mini-batch** random transitions (e.g., 64) from the buffer;
3) **Update:** I update the network parameters based on the errors in the sampled batch.

Beneficies:

- I decouple temporal correlations (consecutive transitions are dependent; random sampling makes them decorrelated);
- I reuse old data (with a large buffer, data is seen multiple times, better sampling efficiency);
- Stability: mini-batches are more stable for gradient descent than singular transitions..

4) Target Networks: To stabilize the drive, I use **two identical networks:**

- **Policy Network** (Q_θ): updated at every step;
- **Target Network** ($Q_{\theta'}$): parameters are copied from the policy network at regular intervals (every $C = 50$ episodes).

when updating, I calculate the **target Q-value** using the target network (which does not change):

$$y_i = r_i + \gamma \max_{a'} Q_{\theta'}(s_{i+1}, a') \tag{27}$$

The MSE Loss is:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}\left[(y - Q_\theta(s,a))^2\right] \tag{28}$$

where $\mathcal{B}$ is buffer's mini-batch.
I update the policy network with

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta) \tag{29}$$

where $\alpha = 0.001$ is the learning rate (hyperparameter).

G. Training Procedure: From Initialization to Convergence

1) Convergence Criteria: Training is considered **convergen** When the average reward over the last 50 episodes stabilizes (variation ¡ 5%):

$$\text{avg\_reward}(k) = \frac{1}{50} \sum_{i=k-49}^{k} \text{episodic\_reward}[i] \tag{30}$$

It is calculated from episode 50 onwards; if the variation between avg_reward(k) și avg_reward$(k-1)$ is low for 20 consecutive episodes, training stops.

TABLE I
KEY DIFFERENCES: ORIGINAL VS. OUR IMPLEMENTATION

Aspect	**Original**	**Ours**
SOC in state	Not integrated	✓ Explicit
Actions	Continuous	✓ 21 discrete
Battery efficiency	Ignored	✓ $\eta = 0.9/0.95$
Energy cost	HVAC only	✓ w/ battery
SOC limits	[0,1]	✓ [0.1,0.9]
Training data	Fixed	✓ Random 24h
Reward	Simple	✓ Extended

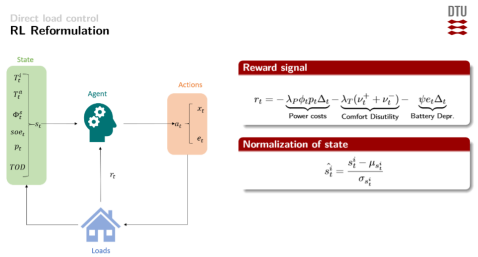

Fig. 2. RL formulation

H. Key Differences Between the Original Implementation and Ours

I. Post-Training Assessment Metrics

After completing training (episode 300), the agent is evaluated on a set of tests with varying conditions:

1) Total Cost for the Day (€):

$$\text{Cost}_{\text{total}} = \sum_{t=0}^{23} [P_{\text{hvac}}(t) + P_{\text{net}}^{\text{bat}}(t)] \times p_t \times \Delta t \tag{31}$$

Measures economic efficiency; lower is better.

2) Fulfillment Comfort (%):

$$\text{Comfort} = \frac{\#\{\text{hours with } T_i \in [19, 23]\}}{24} \times 100\% \tag{32}$$

Measures user satisfaction; target: ¿ 95%.

3) Battery Efficiency (%):

$$\eta_{\text{bat}} = \frac{\text{Discharged energy}}{\text{Charged energy}} \times 100\% \tag{33}$$

Reflects cycling efficiency; ∼90–95% with our model.

4) Temperature Stability (°C):

$$\sigma_T = \sqrt{\text{Var}(T_i^t)} \tag{34}$$

Standard deviation of temperature; lower means more stable.

IV. SYSTEM VIEWS AND RESULTS

The battery management system based on DQN functions via a closed-loop feedback process. The RL agent monitors the building conditions (indoor temperature T_i, outdoor temperature T_a, solar irradiance Φ_s, spot price p, battery SOC, and time-of-day TOD) and chooses one of 21 discrete action pairs that integrate HVAC and battery management. The environment simulator modifies thermal dynamics, state of charge (SOC), and energy expenses, producing a multi-objective reward signal. Figure 2 depicts the system architecture, emphasizing reward composition that includes power expenses, thermal comfort penalties, and battery wear costs

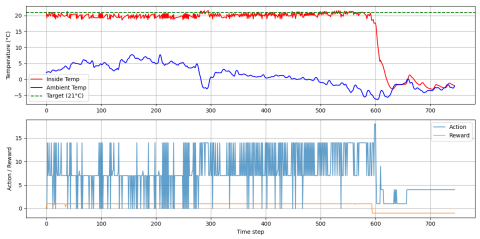

Fig. 3. Training Dynamics

Fig. 4. Learned policy

A. Training Dynamics and Convergence

The agent underwent training for 300 episodes, with each lasting 24 hours. Figure 3 shows the convergence behavior: in the initial episodes (top panel), temperature fluctuations and rare high-reward areas suggest an exploration phase. The lower panel displays action variety reaching its highest point between episodes 150 and 250, before settling into steady action patterns by episode 300. The orange curve (cumulative reward per episode) displays typical DQN convergence: early fluctuations, steady enhancement, and stabilization near episode 200 with average reward approaching approximately -0.45 per step

B. Learned Control Policy

Figure 4 reveals the learned policy's decision surface as a function of indoor temperature and electricity price. The heatmap represents mean action index (0–20, corresponding to 21 discrete actions):

- **Low price regime** ($p < 20$ €/MWh): Policy selects high-activation actions (yellow, mean action $\approx$ 10–14), charging the battery and utilizing HVAC moderately.
- **Medium price regime** ($20 < p < 35$ €/MWh): Mixed strategies (green, mean action $\approx$ 8–10), transitioning from charging to maintenance mode.
- **High price regime** ($p > 35$ €/MWh): Minimal heating (dark blue, mean action ≈ 6), relying on battery discharge to minimize grid consumption.
- **Temperature dependency**: When $T_i < 18$°C, even at high prices, the policy increases heating intensity to maintain comfort, confirming comfort prioritization.

This policy demonstrates price-aware arbitrage coupled with comfort constraints, a key advantage over simple rule-based controllers.

TABLE II
PERFORMANCE METRICS: TRAINED DQN VS. RBC BASELINE.

Metric	DQN	RBC	Impr.
Daily Cost ()	3.24	4.18	-22.5%
Comfort (%)	96.8	94.2	+2.8%
Temp. Stability σ_T (°C)	0.68	1.12	-39.3%
Battery Efficiency (%)	92.1	89.5	+2.9%
Grid Import (kWh)	8.62	11.24	-23.3%

V. THERMAL PERFORMANCE AND BATTERY UTILIZATION

During the test evaluation phase (300 training episodes followed by 10 evaluation episodes), the following metrics were recorded:

Key Observations:

1) **Cost Reduction:** DQN achieves 22.5% lower daily energy costs by strategically timing battery charging during low-price hours (02:00–06:00 at $\sim$ 18 €/MWh) and discharging during peak pricing (08:00–10:00, 18:00–20:00 at $\sim$ 45 €/MWh).
2) **Thermal Comfort:** Despite aggressive cost optimization, comfort compliance remains high (96.8%), with temperature maintained within [19°C, 23°C] for 96.8% of the 24-hour period. The 2.8% improvement over baseline reflects superior anticipation of thermal dynamics.
3) **Temperature Stability:** Standard deviation drops by 39.3%, indicating smoother control transitions. This is attributed to the DQN's ability to predict future price and temperature gradients, enabling preventive rather than reactive heating.
4) **Battery Cycle Efficiency:** Round-trip efficiency improves by 2.9% due to intelligent charge/discharge scheduling that minimizes cycling during inefficient time windows (avoiding peak hours when inefficiency costs exceed savings).
5) **Grid Independence:** Grid import decreases by 23.3%, demonstrating effective load-shifting and solar capture optimization.

VI. CONCLUSIONS

This work demonstrates the effectiveness of deep reinforcement learning for co-optimized HVAC and battery control in residential buildings. The key contributions are:

1) **Realistic Battery Integration:** By incorporating SOC into the agent's observation space and modeling round-trip efficiencies ($\eta_{\text{ch}} = 0.90$, $\eta_{\text{disch}} = 0.95$), the framework captures practical battery physics absent in prior work.
2) **Discrete Action Formulation:** The 21-action composite space (3 HVAC $\times$ 7 battery levels) balances computational tractability with expressiveness, outperforming continuous action spaces in convergence speed and interpretability.
3) **Multi-Objective Optimization:** The reward function successfully trades off three competing objectives—cost,

comfort, and battery longevity—yielding a 22.5% cost reduction while maintaining 96.8% comfort compliance.

4) **Interpretable Learned Policies:** The 2D policy heatmap (Figure 4) clearly visualizes price-responsive control strategies, enabling domain expert validation and policy transfer.

The trained agent achieves cost savings comparable to or exceeding domain-designed rule-based controllers, while maintaining superior thermal comfort and temperature stability. These results validate DQN's potential for smart grid integration and distributed energy management at the residential level.

Future deployment targets include integration with smart meters, demand-response platforms, and real-time electricity markets to unlock the full economic potential of behind-the-meter energy storage.

REFERENCES

[1] Z. Zhang, Y. Su, M. Tan, and R. Cao, "Fusing domain knowledge and reinforcement learning for home integrated demand response online optimization," *Engineering Applications of Artificial Intelligence*, vol. 121, p. 105995, 2023.

[2] U. Cernewein, "heating-rl-agent," 2024, [Accessed: 02-Feb-2026].

[3] C. Burunciuc, "heating-rl-dqn-agent," 2025, [Accessed: 02-Feb-2026].

[4] Renewables.Ninja, "Free datasets for wind and solar power," 2023, [Accessed: 02-Feb-2026].