Master’s Thesis

Adaptable Building Energy Management Systems through Meta Reinforcement Learning

Comparative Study of Standard RL & Meta-RL Families in Energy Domain

In partial fulfillment of the requirements for the degree
Master of Science (M.Sc.)
at the TUM School of Engineering and Design

Examiner	Prof. Dr. Thomas Hamacher Chair of Renewable and Sustainable Energy Systems
Advisor	Dr.-Ing. Ulrich Ludolfinger Chair of Renewable and Sustainable Energy Systems
Submitted by	Ahmed El-Essawy
Submitted on	Munich, February, 1, 2026

Statement of Academic Integrity

I,

Last name: El-Essawy
First name: Ahmed
ID No.: 03781960

hereby confirm that the attached thesis,

Adaptable Building Energy Management Systems through Meta Reinforcement Learning

was written independently by me without the use of any sources or aids beyond those cited, and all passages and ideas taken from other sources are indicated in the text and given the corresponding citation.
I confirm to respect the “Code of Conduct for Safeguarding Good Academic Practice and Procedures in Cases of Academic Misconduct at Technische Universität München, 2015”, as can be read on the website of the Equal Opportunity Office of TUM.
Tools provided by the chair and its staff, such as models or programs, are also listed. These tools are property of the institute or of the individual staff member. I will not use them for any work beyond the attached thesis or make them available to third parties.

I agree to the further use of my work and its results (including programs produced and methods used) for research and instructional purposes.

I have not previously submitted this thesis for academic credit.

Munich, February, 1, 2026 ____________________

Abstract

The growing deployment of distributed energy resources (DERs), including photovoltaic (PV) generation, behind-the-meter battery storage, and flexible electrical loads, has materially intensified the operational complexity of Building Energy Management Systems (BEMS) which must be designed to accommodate heterogeneous building configurations, such as differences in PV sizing, storage capacity, and occupancy-driven demand profiles, while simultaneously addressing multiple objectives. These objectives typically include minimizing energy expenditure and associated emissions, maintaining occupant comfort, and/or mitigating battery degradation. Moreover, policy and regulatory initiatives are increasingly incentivizing the provision of grid-oriented services from residential DERs, including flexibility and demand response, thereby expanding the functional requirements imposed on BEMS.

Despite these evolving requirements, rule-based controllers (RBC) remain the dominant approach in practical deployments. However, RBC performance is typically constrained by limited capability to optimize long-term objectives, reduced robustness across diverse building characteristics and tariff structures, and a reliance on manual retuning for new installations. Model Predictive Control (MPC) provides a systematic optimization framework; nevertheless, its practical scalability is often strongly dependent on accurate forecasts and the need for detailed, building-specific model calibration, which jointly render deployment across heterogeneous building portfolios labor-intensive.

Reinforcement Learning (RL) has consequently been investigated as an alternative solution, as control policies can be learned directly from data and may attain performance comparable to MPC without requiring explicit, detailed building models. However, conventional RL approaches frequently remain limited by data efficiency and generalization when exposed to previously unseen operating conditions. They can overfit to the training environment (dynamics, stochasticity, initial-state distribution, reward quirks), leading to brittle behavior under even modest distribution shifts.

Meta-Reinforcement Learning (Meta-RL) leverages task priors from diverse environments to foster fast adaptability to new buildings and operating conditions with minimal exploration. This thesis explores the potential of Meta-RL not only to enhance adaptability but also to improve data efficiency and generalization in BEMS. Within this framework, the zero-shot performance of conventional deep RL is evaluated alongside Meta-RL under varying environmental conditions and building configurations. The comparative assessment focuses on minimizing energy costs alongside other key performance indicators, including carbon emissions and grid-interaction metrics. The results indicate that Meta-RL can improve the scalability and responsiveness of BEMS in dynamic, real-world energy settings.

keywords - **Building Energy Management Systems, Distributed Energy Resources, Meta Reinforcement Learning, Adaptability, Generalization, Data Efficiency**

Contents

List of Figures

List of Tables

List of Abbreviations

BEMS Building Energy Management Systems. iii, vii, viii, 1–4, 6, 8, 10–13, 20, 21, 23–26, 29–31, 58, 59, 68

BPTT Backpropagation Through Time. 47

DDPG Deep Deterministic Policy Gradient. 20, 24

DER Distributed Energy Resource. iii, 6, 26

DQN Deep Q-Network. 9

ESU Energy Storage Unit. 3, 21, 25, 27, 29, 33, 50

EV Electric Vehicles. 9

FIFO First In First Out. 42

GRU Gated Recurrent Unit. 45, 47

KL Kullback-Leibler. 17, 36, 37

KPI Key Performance Indicator. v, vii, viii, 4, 54, 55, 57, 60, 64, 65, 67

MAML Model Agnostic Meta Learning. 15, 19, 21, 23, 24

MANN Memory-Augmented Neural Network. 40, 57, 60, 61, 63, 67

MDP Markov Decision Process. vii, 3, 4, 9, 14, 15, 25, 30

Meta-RL Meta Reinforcement Learning. iii, v, vii, viii, xi, 2–4, 13–16, 18–21, 23, 24, 29, 35, 36, 40, 43–45, 48, 49, 53, 55–64, 67, 68, 70–75

MPC Model Predictive Controllers. iii, 1, 4, 6–9, 21, 24

PEARL Probabilistic Embeddings for Actor-Critic Reinforcement Learning. viii, 16, 17, 19, 20, 24, 36, 57, 60, 61, 63, 64, 67

POMDP Partially Observable Markov Decision Process. 30

PPO Proximal Policy Optimization. vii, viii, 2, 4, 9–13, 20, 23, 24

PV Photovoltaics. iii, v, 3, 9, 25, 28, 50

RBC Rule-Based Controllers. iii, 1, 4, 6, 7, 9, 21, 24, 53, 54, 57, 60, 64, 65, 67

RL Reinforcement Learning. iii, vii, 1, 2, 4, 8–11, 13, 14, 16, 17, 20–22, 24, 26, 31, 54, 57, 64, 67, 68, 71

RLC Reinforcement Learning Controllers. 9

SAC Soft Actor-Critic. vii, viii, 2, 4, 9, 11–13, 16, 17, 20, 21, 24, 36, 37, 40–42, 46, 53, 57–64, 67, 70, 71

SOC State of Charge. vii, 3, 25, 26, 50–52, 75

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Energy management in the building sector constitutes a challenging control problem due to uncertainty and the coexistence of numerous controllable and uncontrollable factors. In particular, variability in renewable generation, fluctuations in energy prices, environmental disturbances, and operational constraints collectively increase system complexity. As a result, effective operation necessitates robust control strategies that can maintain performance under dynamic and uncertain conditions.

Rule-Based Controllers (RBC) and Model Predictive Controllers (MPC) have been widely adopted for Building Energy Management Systems (BEMS). Relative to conventional RBC-based approaches, MPC can yield improved performance by computing decisions that are optimised with respect to forecast information. However, MPC performance is strongly contingent on forecast accuracy and on the fidelity of the underlying energy system model [32]. Moreover, because a mathematical optimisation problem must be solved at each control step, computational demands may become substantial as the number of constraints, variables, and parameters increases [17]. By contrast, RBC methods are typically characterised by low implementation complexity and computational efficiency, as decisions are generated via predefined rules derived from historical data and current observations [29].

Reinforcement Learning (RL) has therefore attracted increasing attention for energy management, primarily because policies can be learned directly from observed system states through trial and error without requiring an explicit system model. During training, the effective dynamics are implicitly captured through interaction, which can mitigate the impact of modelling inaccuracies that often degrade MPC performance [38]. Furthermore, once training has been completed, RL policies can produce control actions with substantially lower online computational burden than MPC-based approaches [17]. Prior studies have also indicated that RL can outperform RBC methods, particularly in highly dynamic environments where fixed-rule logic is insufficient [33].

However, Conventional deep RL methods such as PPO and SAC are often limited by data efficiency and generalization. PPO is explicitly on-policy, so it must continually collect fresh trajectories and largely discards old data, which makes it comparatively sample-inefficient for problems where interaction is expensive or slow [47]. SAC improves data reuse via off-policy learning and replay buffers, but in practice, it still typically needs very large numbers of environment steps and careful reward/constraint design to reach stable performance in continuous control settings [21]. In addition, deep RL policies can overfit to the training environment (dynamics, stochasticity, initial-state distribution, reward quirks), leading to brittle behavior under even modest distribution shift; this has been demonstrated empirically in studies on overfitting and evaluation methodology [23][55] and in benchmarks designed to measure train–test generalization gaps across procedurally generated or held-out conditions [6]. Together, these results motivate the view that "good training returns" for PPO/SAC do not necessarily imply (i) feasible learning costs in real deployments or (ii) robust performance outside the training distribution.

In BEMS, these same limitations become more acute because each interaction is with safety-critical systems and in real time (minutes-to-hours control intervals), making naïve exploration and the millions of steps often used in PPO/SAC studies impractical; consequently, much of the literature relies on simulators or historical batch data and highlights data hunger as a barrier to deployment [51]. Recent studies [58][14] show that an RL agent may require up to 5 million interaction steps to match the performance of a feedback controller for an HVAC system. The generalization problem is also repeatedly observed: learned policies are frequently building-specific. A controller trained on one building typically requires retraining or substantial retuning to work well in another, as envelope properties, battery sizing, occupancy/usage patterns, and weather conditions induce a strong domain shift. This "train-on-one-building, re-train-on-the-next" reality is emphasized in survey evidence from building-control RL [51].

1.2 Objective of Study

This thesis aims to address the aforementioned issues by answering the following key question:

> **To what extent could zero-shot generalization be achieved together with Meta-RL adaptability so that the resulting controller remains robust under distribution shifts (e.g., new buildings, climates, tariffs, and operating conditions) without requiring extensive post-deployment retraining or fine-tuning?**

This question is important because real-world BEMS deployments impose strict practical constraints. Operating conditions vary substantially across buildings, and controllers must remain safe and feasible immediately after deployment. Therefore, strengthening generalization alongside adaptability increases the likelihood that a Meta-RL controller can be deployed “out of the box” and still maintain strong performance under realistic distribution shifts.

1.3 Scope of Work

This master’s thesis addresses deployment-oriented control for a BEMS that coordinates a battery energy storage system (ESU), PV generation, and non-shiftable electrical loads. In contrast to model-based supervisory control, this work assumes that no explicit, trusted parametric model of building dynamics or exogenous processes is available; moreover, the controller is not provided with the probability distributions governing demand, PV output, prices, or weather-related drivers.

The control objective is to minimize the cumulative electricity bill over a finite horizon, where the agent must learn cost-effective grid-exchange strategies that balance immediate tariff signals against future opportunities and constraints induced by storage dynamics.

Beyond cost optimality, the problem is explicitly framed around practical deployment criteria: the policy should learn from limited interaction data (sample efficiency), train within realistic compute budgets (wall-clock efficiency), and remain robust when transferred across heterogeneous buildings and operating regimes (generalization under distribution shift).

To represent heterogeneity across buildings and operating conditions, the control problem is cast as a family of Markov decision processes (MDPs), where each task corresponds to a specific building scenario sampled from an underlying task distribution. Tasks differ along dimensions that matter in practice, such as (i) demand magnitude and temporal profile, (ii) PV sizing and intermittency, (iii) ESU capacity, power limits, and efficiencies, (iv) climate zone and weather realizations, (v) time-varying electricity prices, and (vi) initial SOC.

The thesis designs and evaluates three actor-critic based Meta-RL algorithms for BEMS, using a common meta-learning framework in which training across a distribution of building tasks enables transferable knowledge (e.g., initialization, context, memory) and fast adaptation to new environments.

The meta-training procedure is organized into the following steps:

1. Task distribution sampling. A batch of buildings/tasks is sampled each iteration.

2. Instead of extremely long annual rollouts, training uses one-week episodes sampled at randomized start times across the year.

3. Building-level interaction and task-local adaptation. Each building maintains its own experience buffers and adaptation state, updated from task-specific data.
4. Meta-level update and cross-task transfer. The meta-learner updates the shared actor–critic backbone and the algorithm-specific meta component using aggregated learning signals.
5. Parallel rollout collection. Training is engineered for throughput by executing multiple environment instances concurrently and batching learning signals for vectorized updates.

The thesis delivers a unified evaluation of three actor–critic Meta-RL variants for BEMS, designed to be deployed “out of the box” and still maintain strong performance under realistic distribution shifts.

The key outcomes and contributions of this work are summarised as follows:

- Benchmarking three Meta-RL algorithms on unseen buildings, including zero-shot generalization and fast adaptation.
- Development and formalization of a shared training framework targeting sample efficiency and wall-clock efficiency.
- Comparison between standard RL baselines and Meta-RL agents under identical evaluation conditions.
- Meta-RL performance evaluated using financial and grid-interaction KPIs.

1.4 Structure of Work

Chapter 2 provides a technical overview of state-of-the-art BEMS controllers such as RBC, MPC, PPO, and SAC while also discussing their technical limitations, and finally presents a brief technical overview of Meta-RL. Chapter 3 reviews recent Meta-RL studies with an emphasis on their reported limitations, which motivate the contributions of this thesis. Chapter 4 will outline the problem formulation addressed in this study. It will begin by detailing the environment setup used for simulations and then progress to MDP formulation.

Chapter 5 details the common actor–critic Meta-RL framework and the three proposed Meta-RL variants, including the meta-training and meta-testing procedures across building tasks. Chapter 6 constructs the evaluation schema for the trained Meta-RL agent and explains how to benchmark its zero-shot deployment against selected baseline controllers using financial and grid-interaction KPIs. Chapter 7 reports and discusses the empirical findings, comparing the proposed algorithm against the selected baselines. Lastly, Chapter 8 concludes the thesis with a summary of the key findings and directions for future research.

Chapter 2

Background and Fundamentals

2.1 Building Energy Management Systems: Context and State of the Art

2.1.1 Germany's Role in the EU 2050 Climate Agenda

In the first quarter of 2025, the EU economy's greenhouse gas emissions increased by 3.4% compared to the same period in 2024, reaching around 900 million tonnes of CO2-eq. Out of the total greenhouse gas emissions, the electricity and gas supply sector alone contributed 19.3% [40]. To address such high levels of greenhouse gas emissions, the EU has committed to becoming climate-neutral by 2050, achieving net-zero greenhouse gas emissions. To realise this goal, each EU Member State prepared a 10-year plan for the period 2021 to 2030. Amongst those states is Germany, which has set a target of increasing the share of renewables in the gross electricity consumption up to at least 80% by 2030 [26].

Furthermore, renewable energy sources are often associated with volatility and strong dependency on weather conditions. This variability affects factors like the electricity supply reliability and stability, heightening the need for energy storage solutions and sector coupling technologies, which aim to use the energy from these sources for other purposes, such as heating or mobility. For instance, by the end of 2024, the total capacity of battery storage systems in Germany had grown to 19 GWh [25] to support the integration of renewable resources. On the demand side, flexibility has become more common in order to support the grid, reduce peak electricity demands, and make use of different electricity price signals.

2.1.2 Strategic Role and Market Growth of Building Energy Management Systems

To meet the increasingly ambitious climate and energy targets established by Germany and the European Union, the deployment of BEMS has become a strategic priority. BEMS enable the real-time monitoring, control, and optimisation of building-level energy flows, thereby providing the digital infrastructure required for the integration of DERs, the improvement of energy efficiency, and the provision of demand-side flexibility. In 2023, the global BEMS market was estimated at approximately USD 6.5 billion, with Europe accounting for more than 30% of total market value [36]. In Germany, BEMS adoption is projected to grow at a compound annual growth rate (CAGR) exceeding 11% over the period 2024–2030, driven by regulatory requirements, ongoing smart grid expansion, and targeted funding mechanisms such as the Bundesförderung für effiziente Gebäude (BEG) [8]. Correspondingly, investment in smart building technologies, including BEMS, is expected to surpass €25 billion by 2030 [42], indicating sustained political and market commitment to digitalisation within the built environment.

The relevance of BEMS is further reinforced by structural shifts in the energy system that are expected to increase both the magnitude and complexity of building-level electricity demand. By 2030, residential electricity consumption in Germany is projected to rise by nearly 30%, primarily attributable to the electrification of heating through technologies such as heat pumps and the expansion of electric mobility, including residential electric vehicle charging [2]. In parallel, more than 60% of new residential buildings are anticipated to incorporate on-site renewable generation, predominantly photovoltaic systems, with an increasing share complemented by battery storage and electric vehicle infrastructure [15]. These developments necessitate intelligent coordination across generation, storage, and consumption, which constitutes a core functional role of BEMS. Moreover, behind-the-meter DER capacity in German buildings is expected to reach 30–40 GW by 2030 [27], thereby underscoring the need for scalable and interoperable BEMS platforms capable of managing increasingly decentralised and heterogeneous energy systems.

2.1.3 Predominance of Rule-Based and Model Predictive Controllers

In present-day BEMS deployments, supervisory control is most commonly implemented through either RBC or MPC paradigms. A review by Amasyali and El-Gohary (2018) reported that approximately 65–75% of commercial and residential energy management systems employ RBC approaches, largely attributable to their conceptual simplicity, modest computational requirements, and straightforward integration into existing automation architectures [4]. By contrast, MPC has attracted increasing attention, particularly within research settings and high-performance building applications, where its deployment has been estimated to account for roughly 15–25% of advanced smart building implementations [30][1]. Nevertheless, RBC remains the prevailing choice in practice, especially for legacy installations, residential contexts, and small-scale commercial buildings, where cost constraints and implementation complexity continue to limit broader MPC adoption.

1. Rule-Based Controllers

RBC schemes are characterised by predefined decision logic in which system responses are governed by manually specified “if–then” rules. Control actions are typically triggered by observable conditions such as indoor temperature, occupancy states, or electricity price signals. Rule sets are generally developed by domain experts, commissioning engineers, or facility managers, and are not derived through formal optimisation or data-driven learning procedures.

RBC is widely used because:

- **Implementation effort** is comparatively low and the resulting control logic is readily interpretable.

- **Computational requirements** are minimal, rendering RBC suitable for embedded hardware and cost-sensitive applications.

- The deterministic nature of rule execution further facilitates verification, commissioning, and troubleshooting in operational environments.

However, the suitability of RBC has been increasingly constrained by the growing complexity and variability of modern building energy systems.

- RBC does not provide an explicit mechanism for optimising long-term objectives such as cost minimisation, comfort maximisation, or emissions reduction, which can lead to systematically suboptimal operation under time-varying conditions [39].

- Rule sets are commonly tailored to a particular building configuration, tariff structure, and occupant profile; consequently, changes in building usage, equipment characteristics, or market rules typically require manual redesign and re-tuning [24].

- RBC exhibits limited adaptability to dynamic pricing, high penetrations of variable renewable generation, and heterogeneous or uncertain occupant behaviour [12].

- For large portfolios of buildings, manually designing and tuning rule sets becomes infeasible.

2. Model Predictive Controllers

MPC constitutes an optimisation-based control strategy in which a predictive model of building energy dynamics is used to forecast future system states and compute control trajectories that minimise a specified objective function over a finite horizon. At each control step, the optimisation problem is solved using updated measurements and forecasts (e.g., weather, occupancy, and electricity prices), and only the first control action is applied before the horizon is shifted forward in a receding-horizon manner.

MPC enables systematic optimisation of operational objectives and can be configured to minimise energy costs, improve thermal comfort, or reduce emissions subject to constraints on equipment operation and indoor environmental quality [9]. Its explicit use of forecasts allows proactive decision-making, which is particularly relevant under variable renewable generation and time-varying tariffs. Furthermore, MPC naturally accommodates multi-objective formulations, enabling trade-off management between competing goals such as comfort and cost.

Despite these benefits, several barriers continue to limit the scalability and widespread deployment of MPC in mainstream BEMS

- The design of an MPC controller typically requires substantial expertise to specify appropriate models, constraints, and objective functions, which restricts applicability in consumer-grade or resource-constrained settings [16].
- predictive models often require building-specific identification and calibration, as thermal and operational dynamics differ significantly across building typologies, construction characteristics, and usage patterns [7].
- The resulting development and maintenance burden can be considerable for heterogeneous building portfolios, particularly when data availability and quality are inconsistent and when model drift arises due to equipment ageing or behavioural change [48].
- real-time MPC can impose nontrivial computational demands, which may be prohibitive for low-cost controllers or systems with strict latency requirements unless simplifications or dedicated computation resources are introduced [35].

2.2 Reinforcement-Learning for Building Energy Management Systems

RL is a control paradigm in which an agent learns a policy (a mapping from states to actions) by interacting with an environment and receiving rewards that encode the control objective as seen in Figure 2.1 [49]. In building control, RL is attractive because model-free RL can, in principle, learn effective control policies without assuming an explicit physics model of the building's dynamics; instead, it improves behavior through trial-and-error experience (often in simulation first) using signals such as indoor temperature, weather, occupancy proxies, electricity price, PV generation, and device states. Common deep RL variants used in buildings include value-based methods (e.g., DQN) and policy-gradient / actor–critic methods suited for continuous control (e.g., PPO, SAC).

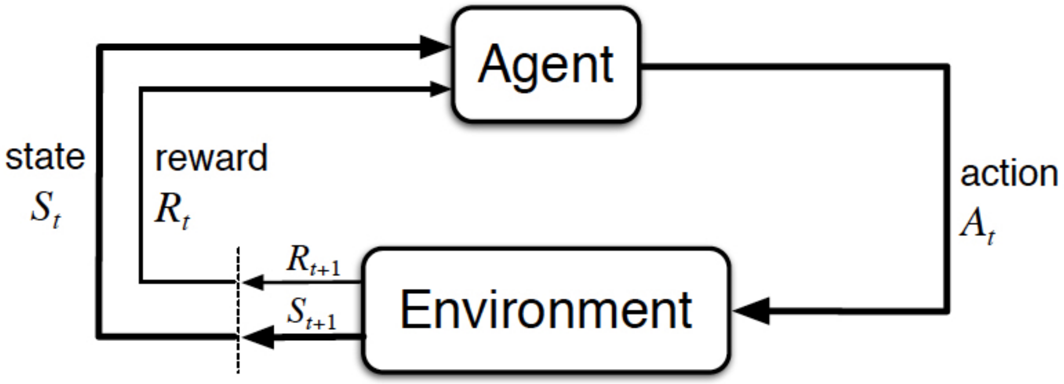


Figure 2.1 The agent–environment interaction in a Markov Decision Process (MDP) [49].

RLC have gained momentum in building energy control primarily because they mitigate several practical limitations of RBC and MPC. Present-day operating environments are characterised by dynamic tariffs, carbon-intensity signals, high PV penetration, EV charging, widespread adoption of heat pumps, and the integration of batteries. Collectively, these developments yield non-stationary and strongly coupled control problems that are:

- difficult to represent with static heuristic rules and
- costly to model and calibrate for MPC at scale. [52][10]

Consequently, RLC are frequently positioned as a data-driven alternative that can learn policies directly from interaction with the environment, thereby reducing reliance on hand-engineered rules and explicit physics models.

2.2.1 Policy–Gradient and Actor–Critic Methods in Brief

Sutoon & Barto explain in [49] that Policy-gradient RL optimizes a *parameterized policy* $\pi_\theta(a \mid s)$ directly by ascending the gradient of expected return. Actor-critic methods split the job into:

- an actor (the policy π_θ) that proposes actions, and

- a critic (a value function $V_\varphi(s)$ or action-value $Q_\varphi(s,a)$) that estimates how good actions/states are, providing lower-variance learning signals than pure Monte Carlo policy gradients.

2.2.2 Algorithm 1: Proximal Policy Optimization

The outcome of the agent–environment interaction (reward/cost + next state), which is referred to as "Experience" in Figure 2.2 [47][50], is collected into the **Trajectory**. Using the stored experience, the critic updates itself to better judge which actions were good/bad in each situation. The actor is updated to favor actions the critic says were better and avoid actions the critic says were worse. PPO's key idea is: when improving the actor, don't let the policy change drastically in one training step. So PPO compares the new actor to the old actor that collected the trajectory, and limits how big the update can be. This makes learning much more stable (avoids the actor "overreacting" to one batch of experience).

In one sentence, PPO alternates between:

(A) trying actions to collect trajectories

(B) training a critic to judge those actions

(C) improving the actor, but with a safety brake so the actor doesn't change too abruptly between iterations.

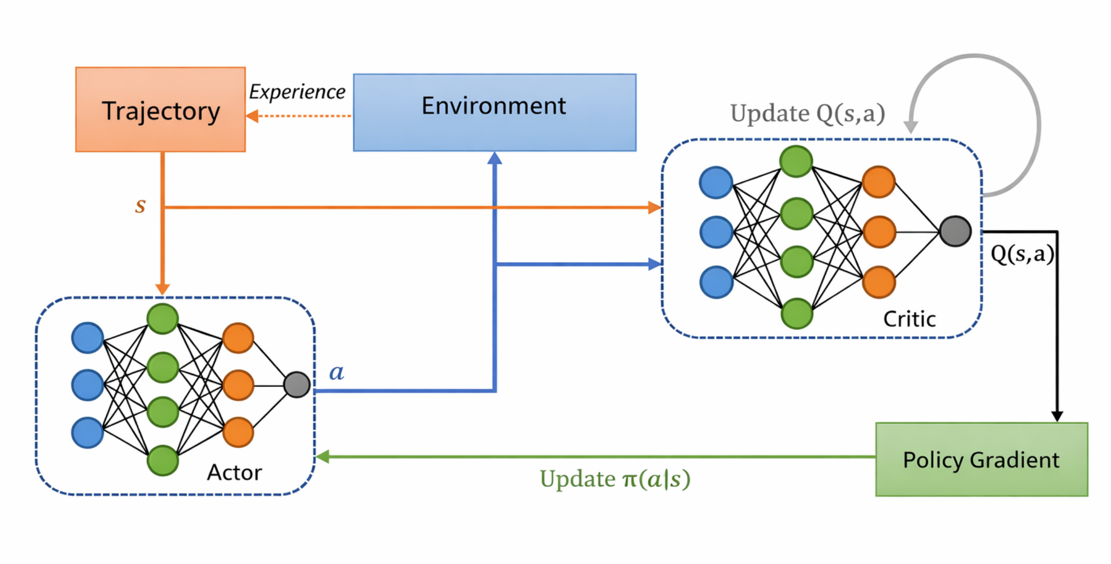


Figure 2.2 Proximal Policy Optimization (PPO) architecture [50].

PPO has been widely adopted as a baseline deep RL method for control tasks; however, several methodological characteristics limit its practical suitability for BEMS:

- **PPO is inherently on-policy**, and previously collected experience is typically discarded once policy updates are performed. As a result, training is sample-inefficient and often requires a large volume of fresh interactions, which is generally acceptable in simulation but can be prohibitively costly or operationally risky in real buildings.[47][23]

- PPO exhibits notable **sensitivity to reward** specification and scaling. In BEMS applications, reward functions frequently encode penalty terms to represent constraint violations. Nevertheless, poorly balanced trade-offs or improperly scaled penalties can destabilize learning dynamics and may induce unintended control behaviors.[23]

- **Constraint satisfaction** is not natively enforced within standard PPO formulations. Consequently, operational constraints related to occupant comfort, equipment limits, and safety are commonly addressed through additional mechanisms such as shielding, constrained reinforcement learning variants, or rule-based supervisory overlays.[23]

- **Generalization** is not guaranteed: like most deep RL, PPO can overfit to a specific simulator, season, tariff, occupancy pattern, or forecast error distribution. [23][55]

2.2.3 Algorithm 2: Soft Actor-Critic

In SAC, the outcome of the agent–environment interaction (reward/cost + next state) is stored in a replay buffer, and then the agent reuses the same past experiences many times for learning. During training, SAC repeatedly samples random mini-batches from this replay buffer to update two main components: a critic (two critics in most SAC implementations) and the actor. The critics learn to predict how good an action is in a state (expected long-term return), using the stored transitions and a stable “target” calculation. Then the actor is trained to choose actions that the critics rate highly.[21]

What makes SAC distinct is its “soft" learning objective: the actor is not only trained to get high reward, but also to remain stochastic (i.e., keep some randomness). In practice, SAC updates the actor to favor actions that have high predicted value while also encouraging exploration through an entropy term. This often produces policies that are smoother and more robust, and it can help the agent avoid getting stuck in narrow behaviors.[21]
In one sentence, SAC alternates between:

(A) collecting transitions and storing them in a replay buffer

(B) training (twin) critics to evaluate actions using reused past experience

(C) improving a stochastic actor to pick high-value actions while staying exploratory.

SAC is frequently selected for continuous control in BEMS due to its off-policy learning and favorable sample efficiency. Nevertheless, several limitations remain:

- SAC introduces greater architectural and algorithmic complexity. More moving parts than PPO: twin critics + targets + entropy increase implementation and tuning complexity. [21]
- **Safety/constraints still not guaranteed**; SAC can explore unsafe actions. This issue is especially critical in occupied buildings, where comfort, equipment limits, and safety constraints must be respected continuously, often necessitating shielding, constrained reinforcement learning extensions, or supervisory rule layers. [23][21]
- **Generalization** across buildings and operating modes remains contingent on the diversity of training data, as well as on the correctness and stability of the deployed state and action interfaces. [55]
- SAC needs nontrivial computational and engineering overhead. Maintaining multiple networks and performing multiple gradient updates per environment step can be burdensome for embedded BEMS hardware or resource-constrained controllers. In practice, this limitation is often mitigated by conducting training offline and deploying only a lightweight policy network. [21][20]

2.2.4 Proximal Policy Optimization vs Soft Actor-Critic

Table 2.1 Comparison of PPO vs SAC

Aspect	PPO [47]	SAC [21]
Data usage	**On-policy:** learns mainly from *fresh trajectories* collected by the current policy, and then moves on.	**Off-policy:** learns from a *replay buffer* and can keep learning from older experiences many times (more sample-efficient).
How policy updates are kept stable	Uses a “safety brake” by explicitly limiting how much the policy can change in one update (clipped update vs the old policy).	Stays stable mainly through *critic-based learning* + *entropy-regularized* (“soft”) objective, rather than comparing against the old policy with a hard update limit.
Exploration style	Often explores via action policies, but it’s not inherently built around maximizing exploration.	Explicitly encourages exploration by rewarding stochasticity (*entropy*), which often helps it adapt and learn efficiently in continuous control.

When selecting an RL algorithm for BEMS, we focus on three practical questions: suitability to the setting, training practicality, and robustness under parameter variation:

(A) Which is most suitable for BEMS?

In many BEMS settings (continuous control, limited real interactions, need to reuse data, frequent parameter variation), SAC is often the more suitable default because it is off-policy and sample-efficient, and the entropy term tends to produce policies that are more robust to noise and minor distribution shifts. [21][20]

(B) Which trains "better" practically?

- **Training stability** (ease of getting a working baseline): often PPO (clean on-policy loop, fewer interacting components). [47][23]
- **Training efficiency** (performance per interaction): often SAC (replay buffer reuse; commonly fewer environment steps to reach a target return in continuous control)[21][20]

(C) Which generalizes better under changing environment parameters?

Neither guarantees generalization. Deep RL is known to be brittle under distribution shift unless trained for it. [55]

- SAC frequently shows better robustness in practice for BEMS-like continuous tasks because Off-policy learning can exploit diverse historical data (if curated) and adapt via continued learning. [21][20]
- PPO can sometimes be more conservative (via clipped updates) and therefore less prone to catastrophic policy jumps, which is useful in safety-critical building operations.[47]

2.3 Meta-Reinforcement Learning

The increasing importance of adaptability to new building dynamics has strengthened interest in Meta-RL. Property managers and aggregators require control strategies that can be transferred or adapted across heterogeneous assets without extensive manual redesign. RL is frequently framed as a pathway toward more automated policy adaptation.

Even if SAC is well-established today, the core unsolved issue in BEMS is fast transfer across many buildings, seasons, tariffs, and equipment configurations. A Meta-RL controller can adapt in a few interactions to a new environment. In the next section, we will understand how Meta-RL could achieve this.

2.3.1 What Meta-Reinforcement Learning is and How it differs from Standard Reinforcement Learning

Standard RL typically optimizes a policy π to maximize expected return in one MDP or in a fixed training environment distribution treated as “one task” [49]. In contrast, Meta-RL optimizes *the ability to learn* across a distribution of tasks, where each task is usually formalized as an MDP $\mathcal{M}$ [13][41][54].

- **Unit of generalization:** Meta-RL generalizes across tasks/MDPs sampled from a distribution $p(\mathcal{M})$, rather than across states within a single MDP [13][41].
- **What is optimized:** Meta-RL learns *meta-parameters* θ that enable rapid adaptation to a new task using limited experience [13][41], or learns an implicit learning algorithm implemented by a recurrent/memory system that updates its behavior based on experience [11][54].

A key motivation is test-time data efficiency: after meta-training, an agent may adapt to a new task with only a few trajectories (few-shot adaptation), because the meta-learner has internalized shared structure across training tasks [13][41]. This advantage is most plausible when (i) test tasks are drawn from the *same or a closely related* distribution as meta-training tasks, and (ii) tasks share transferable structure (e.g., dynamics families, reward motifs, or reusable skills) such that prior experience can constrain exploration and credit assignment [13][41][19]. Meta-RL can fail to deliver data efficiency under substantial distribution shift or when training tasks do not contain exploitable common structure [13][41].

2.3.2 Canonical Meta-training Setup

Let a task be an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel P, reward function r, discount γ, and initial-state distribution ρ_0 [49]. A trajectory is $\tau = (s_0, a_0, r_0, \ldots, s_T)$ sampled by rolling out a stochastic policy $\pi(\cdot \mid s; \varphi)$. Define discounted return

$$R(\tau) \;=\; \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t). \tag{1}$$

For a fixed task $\mathcal{M}$, the RL objective is

$$\mathcal{J}_{\mathcal{M}}(\varphi) \;=\; \mathbb{E}_{\tau\sim\pi(\cdot;\varphi),\,\mathcal{M}}[R(\tau)]. \tag{2}$$

Meta-RL introduces meta-parameters θ and an adaptation operator f_θ that maps experience on a task to adapted parameters φ. A common formulation is:

$$\boxed{\mathcal{J}(\theta) = \mathbb{E}_{\mathcal{M}^i\sim p(\mathcal{M})}\Big[\mathbb{E}_{\tau\sim\pi(\cdot;\varphi_i),\mathcal{M}^i}\big[R(\tau)\big] \;\; \text{s.t.} \;\; \varphi_i = f_\theta(\mathcal{D}_i)\Big]} \tag{3}$$

where $\mathcal{D}_i$ denotes adaptation data (e.g., one/few trajectories) collected in task $\mathcal{M}^i$ using the current behavior, and f_θ may be a gradient step, an inference network, or a recurrent update rule [13][11][41].

Typical meta-training loop: sample a meta-batch of tasks $\{\mathcal{M}^i\}$, collect within-task experience $\mathcal{D}_i$, compute adapted policies φ_i, then evaluate the adapted policies on fresh rollouts for the meta-update; finally, report generalization on held-out tasks from $p(\mathcal{M})$ [13][41][54].

2.3.3 Four Conceptual Families of Meta-Reinforcement Learning

Meta-RL algorithms are classified into families depending mainly on (i) how adaptation happens, or in other words, the mechanism producing task-specific behavior from shared meta-parameters, and (ii) what carries information across tasks/MDPs. Among the prominent families are:

A. Gradient-based Meta-Reinforcement Learning

How adaptation happens: learn an initialization or an update rule such that a few gradient steps on task data produce a high-performing policy for that task (*fast learning via gradients*) [13]. A key algorithm is MAML, which utilizes inner-loop gradient steps to optimize θ for improved post-update performance. [13]. Variants adapt to RL-specific stability constraints by using trust-region or proximal policy updates in the meta-optimization [46][47][43].

What carries information across tasks: the learned initialization θ (and sometimes meta-learned step sizes or update structure) encodes transferable priors about policies and learning dynamics [13][18].

B. Context / latent-variable meta actor-critic

How adaptation happens: infer a latent context variable z summarizing task identity from recent transitions, and condition the policy/value functions on z. Adaptation shifts to probabilistic inference instead of relying on gradient steps, enabling off-policy training and rapid test-time adaptation through updates to the inferred posterior over z [41]. This change redefines the focus of "what adapts" from parameters to beliefs or latent task embeddings.

What carries information across tasks: shared inference and control networks; task-specific information flows through z, estimated from experience [41].

C. Memory-based / episodic Meta-Reinforcement Learning

How adaptation happens: augment agents with external or structured memory so that experience can be stored and retrieved, often via attention or content-based addressing. In meta-learning broadly, memory augmentation supports one/few-shot generalization by retrieving relevant past episodes [44]. In Meta-RL, similar architectural biases appear in sequence models designed for fast adaptation by temporal convolutions and attention (e.g., SNAIL), which have been applied to both supervised meta-learning and RL-style settings [37].

What carries information across tasks: explicit memory contents and retrieval mechanisms provide a fast route for within-task inference and decision-making [44][37].

D. Black-box / recurrent Meta-Reinforcement Learning

How adaptation happens: train a recurrent policy (or agent) end-to-end so that its hidden state implements a learned RL algorithm. The recurrent dynamics integrate past observations, actions, and rewards, producing behavior that improves within an episode or across episodes of the same task [11][54].

What carries information across tasks: recurrent hidden state and learned update dynamics encode how to explore and exploit based on experience [11][54]. These methods can meta-learn exploration strategies because their hidden state can represent uncertainty or hypotheses about the task [54][19].

2.3.4 Probabilistic Embeddings for Actor-critic Reinforcement Learning (Inner Adaptation + Outer Meta-optimization)

A. Core idea

PEARL replaces gradient-based inner-loop adaptation with *probabilistic task inference*. For each task/building $\mathcal{T}_i \sim p(\mathcal{T})$, the agent maintains a *context buffer* C_i (recent transitions) and infers a latent task variable z_i via an amortized context encoder $q_\varphi(z_i \mid C_i)$ [41]. The policy is shared across tasks but conditioned on z, i.e., $\pi_\theta(a \mid s, z)$; fast adaptation at meta-test time is achieved by updating C_i online and re-inferring z_i [41].

PEARL uses a KL regularizer $D_{\mathrm{KL}}(q_\varphi(z_i \mid C_i) \,\|\, p(z))$ (with typically $p(z) = \mathcal{N}(0, I)$) to control the information content of z_i, which can be interpreted through a variational information bottleneck lens [41][3]. Sampling z_i from the posterior supports temporally-extended, structured exploration under task uncertainty [41].

B. Architecture sketch (actor–critic framing)

PEARL is commonly instantiated with an off-policy SAC backbone [21][41]: a shared Gaussian actor $\pi_\theta(a \mid s, z)$, shared twin critics $Q_{\psi_1}(s, a, z)$, $Q_{\psi_2}(s, a, z)$, and corresponding target critics $\bar{Q}_{\psi_1}, \bar{Q}_{\psi_2}$. Task-specific state is carried by (D_i, C_i, z_i), where D_i is a replay buffer and C_i is a context buffer for task i [41].

For each sampled task $i \in \{1, \ldots, N\}$ in a meta-batch:

a. **Initialize buffers and posterior:**

$$D_i \leftarrow \emptyset, \quad C_i \leftarrow \emptyset, \quad q_\varphi(z_i \mid C_i) \approx p(z). \tag{4}$$

b. **Collect transitions (off-policy data):**
Interact with $\mathcal{T}_i$ using $\pi_\theta(a \mid s, z_i)$ to collect transitions $(s_t^i, a_t^i, r_t^i, s_{t+1}^i, d_t^i)$, and store them in D_i. Update the context buffer C_i using recent transitions (e.g., a sliding window) [41].

c. **Infer task latent z_i from context:**
Use the context encoder to form a Gaussian posterior:

$$q_\varphi(z_i \mid C_i) = \mathcal{N}(\mu_i, \mathrm{diag}(\sigma_i^2)), \quad z_i = \mu_i + \sigma_i \odot \varepsilon, \ \varepsilon \sim \mathcal{N}(0, I), \tag{5}$$

where σ_i is the posterior standard deviation (reparameterization) [41][31]. (PEARL implements set-based inference over unordered transitions in C_i [41].)

d. **Critic update (SAC-style, conditioned on z_i):**
Sample an RL batch $B_i^{\text{RL}} \sim D_i$ and a context batch $B_i^{\text{ctx}} \sim C_i$, infer $z_i \sim q_\varphi(z_i \mid B_i^{\text{ctx}})$, and minimize Bellman residuals for $j \in \{1, 2\}$:

$$\begin{gathered} \mathcal{L}_i^Q(\psi_j) = \mathbb{E}\Big[\big(Q_{\psi_j}(s_t^i, a_t^i, z_i) - y_t^i\big)^2\Big], \\ y_t^i = r_t^i + \gamma(1 - d_t^i)\Big(\min_k \bar{Q}_{\psi_k}(s_{t+1}^i, a', z_i) - \alpha \log \pi_\theta(a' \mid s_{t+1}^i, z_i)\Big), \end{gathered} \tag{6}$$

with $a' \sim \pi_\theta(\cdot \mid s_{t+1}^i, z_i)$ and α the SAC temperature [21][41].

e. **Actor update (maximum-entropy objective, conditioned on z_i):**

$$\mathcal{L}_i^\pi(\theta) = \mathbb{E}\Big[\alpha \log \pi_\theta(a \mid s_t^i, z_i) - \min_k Q_{\psi_k}(s_t^i, a, z_i)\Big], \quad a \sim \pi_\theta(\cdot \mid s_t^i, z_i). \tag{7}$$

Optionally tune α toward a target entropy $\mathcal{H}_{\text{target}}$ via:

$$\mathcal{L}_i^\alpha(\alpha) = \mathbb{E}\Big[-\alpha\big(\log \pi_\theta(a \mid s_t^i, z_i) + \mathcal{H}_{\text{target}}\big)\Big] \tag{8}$$

as commonly used with SAC [21].

f. **Encoder (meta-inference) update with KL regularization:**
Train the inference network using gradients flowing through the critic objective, plus a KL penalty:

$$\mathcal{L}_i^\varphi(\varphi) = \mathcal{L}_i^Q(\psi_1) + \mathcal{L}_i^Q(\psi_2) \ + \ \beta\, D_{\mathrm{KL}}\big(q_\varphi(z_i \mid C_i) \,\|\, p(z)\big), \tag{9}$$

where β controls the KL strength [41][3].

g. **Target critic update (Polyak averaging):**

$$\bar{\psi}_j \leftarrow \tau\psi_j + (1-\tau)\bar{\psi}_j, \quad j \in \{1,2\}. \tag{10}$$

h. **Outer meta-update (shared parameters across tasks):**
Aggregate updates over tasks to optimize the meta-parameters $\Theta = \{\theta, \psi, \varphi, \alpha\}$ using off-policy batches from $\{D_i, C_i\}$ [41].

C. How many actors/critics per task?

- **Actors per task:** one *shared* actor $\pi_\theta(a \mid s, z)$ across tasks; task specialization is via z_i inferred from C_i, not via per-task θ_i copies [41].
- **Critics per task:** shared twin critics Q_{ψ_1}, Q_{ψ_2} (and shared targets $\bar{Q}_{\psi_1}, \bar{Q}_{\psi_2}$), all conditioned on z [41][21].
- **Per-task state:** separate D_i and C_i per task/building, and a task posterior $q_\varphi(z_i \mid C_i)$ (parameterized by μ_i, σ_i) [41].

D. Meta-critic interaction

- **Critic-driven representation learning:** the encoder $q_\varphi(z_i \mid C_i)$ is trained using critic losses, so z_i is shaped to capture task information that improves value prediction and control, with KL regularization preventing overfitting to small context sets [41][3].
- **Exploration via posterior sampling of z_i:** sampling z_i from $q_\varphi(z_i \mid C_i)$ and conditioning the policy on it yields temporally coherent exploration under uncertainty; as C_i grows, the posterior concentrates and behavior becomes more task-specialized [41].

2.3.5 Few-shot vs Zero-shot vs Many-shot Meta-Reinforcement Learning

This section presents various evaluation regimes for Meta-RL algorithms deployment. The suitable evaluation regime is determined by the permissible 'test-time data,' or, in other words, the additional on-task interaction after deployment on a held-out task where performance has not yet been evaluated.

A. **Zero-shot**

- **Test-time data:** none
- **What it means:** evaluate the *prior/meta-policy* directly without any adaptation; performance reflects generalization without learning on the new task
- **When it works:** when training has produced a policy that is already robust across tasks or can infer task identity from immediate observations [41].

B. **Few-shot**

- **Test-time data:** small (e.g., 1–N trajectories / episodes)
- **What it means:** evaluate the ability to adapt quickly using limited interaction, either via gradient steps [13] or via fast context inference from a small buffer of experience [41], or via recurrent state updates [11][54].
- **When it works:** when tasks share exploitable structure so that limited data can identify the task and guide efficient improvement [13][41].

C. **Many-shot**

- **Test-time data:** large (substantial interaction)
- **What it means:** evaluate whether meta-training yields a better starting point or learning dynamics that eventually outperform training-from-scratch given enough data [13].
- **When it works:** when continued learning is stable and task shift is not too extreme [13][11].

Which regime best evaluates generalization? Zero-shot is the most direct evaluation of *generalization* because it measures performance on held-out tasks without using additional task-specific data that could blur the line between generalization and within-task learning. Few-shot and many-shot evaluations additionally test *adaptation capability* and *learning efficiency*, respectively, but they confound generalization with the efficacy of the adaptation mechanism.

2.3.6 Meta-Reinforcement Learning in a Nutshell

Meta-RL can be viewed as optimizing a *learning system* over a task distribution via gradient-based fast adaptation (MAML), inference over latent task context (PEARL), or black-box recurrent/memory mechanisms that implement learned update rules [13][41][54][11]. Across these families, the central design question is what internal state is allowed to change at test time (parameters, latent context, hidden state, or memory) and how that choice trades off stability, expressivity, and data efficiency [13][41][54].

Chapter 3

Literature Research

This chapter reviews *implementation and application studies* of Meta-RL, with special focus on its deployment within BEMS. The criteria for including referenced papers are as follows: (i) a clear application of Meta-RL, (ii) empirical validation, (iii) a focus on control, optimization, or adaptive decision-making systems, and (iv) publication in the recent period from 2020 to 2025.

Section 3.2 synthesises comparative findings and provides an overview of the limited number of Meta-RL studies published in recent years. Rather than reiterating Meta-RL's well-established advantage in rapid adaptation, the section focuses primarily on the limitations reported in prior work. These identified gaps form the main motivation for this thesis and clarify the specific contribution "improved zero-shot generalisation to previously unseen environments".

3.1 Resent Meta-Reinforcement Learning Implementations

3.1.1 Meta-RL for Robotic Industrial Control

The study in [45] investigated robotic peg-in-hole insertion tasks, a standard benchmark for precision manipulation, in which each task varied in hole position, shape, and clearance, thereby requiring generalization to previously unseen insertion settings. Meta-RL was selected over conventional deep RL approaches such as DDPG or SAC because these methods typically require retraining for each new task, rendering adaptation computationally and practically costly; in contrast, Meta-RL enables the learning of reusable task priors that support rapid adaptation when novel assembly tasks arise as shown in Figure 3.1.

To this end, the authors used PEARL , a context-based, off-policy Meta-RL algorithm that learns a latent task embedding z from experience and conditions the policy on z. Empirically, the proposed approach reduced adaptation time by approximately 60-80% relative to baseline DDPG and PPO controllers while maintaining strong transfer perfor-

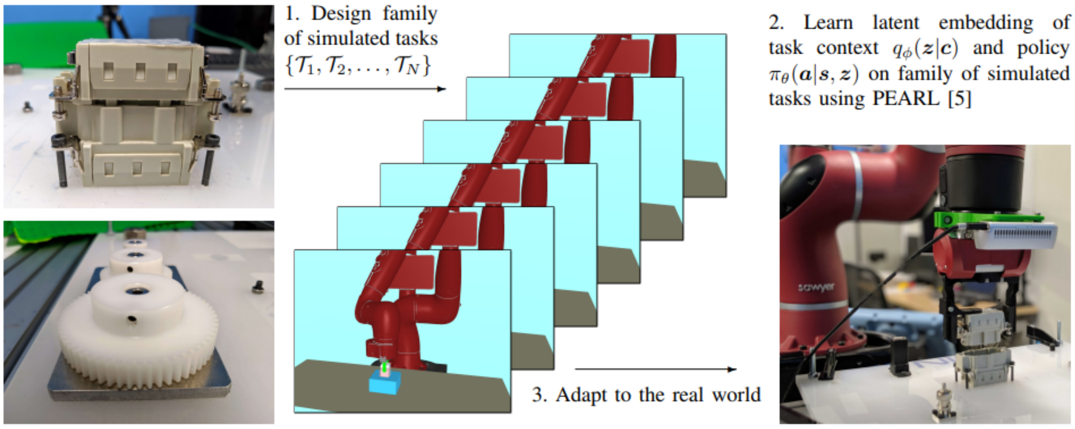

1. Design family of simulated tasks $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$
2. Learn latent embedding of task context $q_\phi(\boldsymbol{z}|\boldsymbol{c})$ and policy $\pi_\theta(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$ on family of simulated tasks using PEARL [5]
3. Adapt to the real world
sawyer

Figure 3.1 Applying meta-learning to two real-world industrial insertion tasks, a waterproof electrical connector plug and a 3D-printed gear [45].

mance across unseen geometries and exhibiting stable convergence under sensor and motor noise. Although the study remains constrained by simulation-to-real discrepancies and a zero-shot success rate of only 20%, its core result, fast adaptation driven by Meta-RL, was achieved.

3.1.2 MetaEMS: A Meta-Reinforcement Learning Framework for Building Energy Management Systems Control

The paper in [56] formulates a learning-based BEMS optimization problem for coordinated control of multiple building components, including an ESU, HVAC modeled as a thermostatically controlled load, renewable generation, and non-shiftable demands across 4 different climate zones. The main objective is to minimize energy cost while shaping load profiles to support grid reliability.

Meta-RL is motivated by practical deployment barriers commonly observed in building control: standard RL-based BEMS approaches typically require extensive interaction data and often fail to adapt robustly to previously unseen buildings. Meta-RL is introduced to transfer experience from previously solved building control tasks to new ones, improving data efficiency and accelerating adaptation.

Concretely, MetaEMS is implemented as an MAML-based actor–critic meta-learning framework with two complementary adaptation mechanisms: (i) building-level adaptation that fine-tunes parameters for each specific building and (ii) group-level adaptation that learns a shared initialization across tasks/buildings. MetaEMS is evaluated against a broad set of baselines, including RBC, SAC with random initialization, pretrained SAC initialization, SAC+MAML, and an RL–MPC hybrid.

The reported results of meta-testing indicate faster adaptation: baseline methods typically converge in roughly 5–7 episodes, depending on the approach, whereas MetaEMS converges within the first three episodes, aligning with qualitative observations of more stable adaptation, as shown in Figure 3.2. These results translate to reduced fluctuations in daily net electricity consumption.

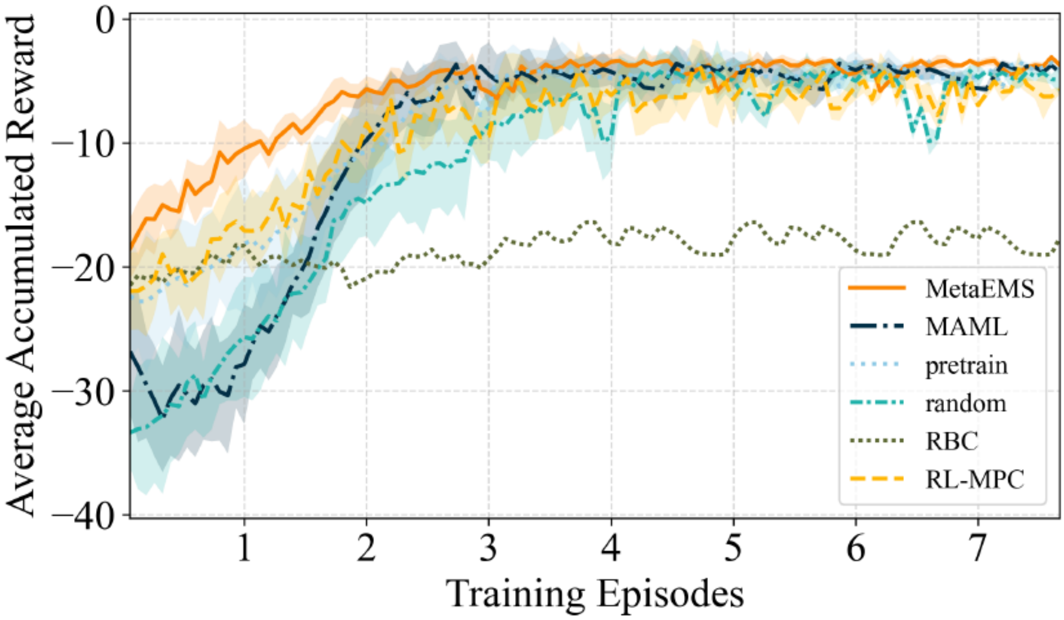


Figure 3.2 Learning curves of MetaEMS vs baseline algorithms. The curve plots accumulated average reward of buildings in one climate zone with means and variances of 5 random seeds [56].

Empirically, the reported results show consistent economic and grid-oriented benefits demonstrated in Table A.1 and Table A.2 respectively.

Nevertheless, as in many simulation-driven studies, a key limitation remains the deployment gap: real-world effectiveness depends on how well the simulated task distribution reflects actual building variability. Unseen-building generalization, which is often evaluated by performance at zero-shot deployment, highlights the ongoing challenge of ensuring robust performance on day one for safety-critical new buildings.

3.1.3 Model Agnostic Meta Learning + Proximal Policy Optimization for Office-building Demand Response Price-setting

The paper in [28] proposes a controller for an office-building demand response experiment in which collecting RL training data is prohibitively expensive. The formulated agent is a price-setting controller that selects prices to influence electricity demand, and it emphasizes the use of simulation to warm-start learning prior to deployment in a real experiment. Meta-RL is motivated by the need for sample efficiency under a simulation-to-experiment mismatch: training a PPO-based controller directly for an hourly pricing task could require months of real-world interaction data, which is impractical for a live office setting, whereas meta-learning over simulated task variations can produce an initialization that adapts rapidly when task conditions shift without relying on a single policy fit to one simulator configuration.

Methodologically, the study adopts MAML applied to a PPO agent (MAML + PPO), using a standard two-layer neural network, with the key rationale that MAML explicitly optimizes for an initialization that can be fine-tuned to new tasks using only a small number of gradient steps.

Quantitatively, the authors report that incorporating MAML reduces total energy cost by roughly 40% on average in both evaluated environments relative to PPO by the end of training, and they further demonstrate a strong warm-start effect: fewer than two simulated days of MAML+PPO training suffices to outperform a PPO baseline trained for 100 simulated days. Qualitatively, they observe that performance improves as the number of MAML training steps increases up to around 200 iterations, after which signs of overfitting to a local minimum emerge.

From a BEMS perspective, the work highlights two key limitations and research opportunities: (i) the need to validate meta-trained controllers under real behavioral uncertainty, and (ii) the importance of integrating Meta-RL with safety, incentive, and operational constraints, given that exploratory pricing and control actions can have nontrivial consequences in real deployments.

3.2 Comparative Analysis

Literature	**Meta-RL for Robotic Industrial Control**	**MetaEMS: A Meta-RL Framework for BEMS Control**	**MAML + PPO for Office-building Demand Response Price-setting**
Domain	Robotics	BEMS	BEMS
Task	Insertion control	HVAC control	Demand Response Price-setting
Meta-RL Algorithm	PEARL	MAML actor-critic	MAML + PPO
Baselines	DDPG, PPO & SAC	RBC, SAC with random initialization, pretrained SAC initialization, SAC+MAML, and an RL–MPC hybrid	PPO
Why Meta-RL	Fast cross-task adaptation	More stable adaptation	Strong warm-start effect
Key Results	60-80% faster adaptation	4-6% cost reduction	40% reduced total energy cost
Key Limitations	Sim-to-real transfer, Extended meta-training duration	Sim-to-real transfer, Unseen-building generalization, and Multi-objective coordination	Sim-to-real transfer, safety & operational exploratory breaches, Extended meta-training duration

Table 3.1 Comparative Analysis of Meta-RL Applications Across Different Domains

Across domains, we can conclude that:

- Meta-RL's strongest value is rapid adaptation under realistic distribution shifts (seasonal changes, occupancy shifts, tariff updates).
- Meta-RL consistently enhances data efficiency which is not just a computational metric; in buildings it translates to fewer risky exploratory actions and less operational disruption.

However, scalability, zero-shot generalization, and wall-clock efficiency remain key barriers to real-world deployment as illustrated above in Table 3.1.

Chapter 4

Problem Formulation

This chapter presents the formulation of the problem considered in this study. Section 4.1 begins by describing the setup of the environment used for the simulations, including an overview of the system components and their interactions. Then it provides a detailed modelling of the individual components, highlighting the key dynamics and constraints relevant to the energy management problem. Finally, Section 4.2 defines the MDP formulation and introduces the base learner function used to estimate a learnable parameter that transfers the learning experience from a single building.

4.1 Environment Overview

This master's thesis investigates sample-efficient, deployment-oriented control methods for BEMS operating under uncertainty and intertemporal feasibility constraints.A typical BEMS architecture comprises several key components, including the building, a control center, a smart meter, electrical loads, an ESU, renewable energy resources, and the connection point to the power grid, as shown in Figure 4.1.
The ESU, represented as a battery pack in this experiment, can reduce net energy drawn from the main grid by locally storing surplus renewable generation PV panels in this case. Additionally, the ESU is subject to operational limits and temporally coupled constraints, including SOC dynamics, charge/discharge power bounds, conversion efficiencies, degradation, and feasibility requirements that link present decisions to future admissible actions. PV generation and load demand are treated as stochastic, time-varying processes. In addition, the BEMS is exposed to time-dependent electricity tariffs and other exogenous signals (e.g., weather proxies) that affect the optimal operating strategy.

A defining assumption of this work is the absence of an explicit, trusted model of building dynamics and exogenous processes. In particular, the controller doesn't require (i) a parametric state-transition model of the building and storage system, and/or (ii) prior knowledge of the probability distributions governing PV output, demand, price, or weather. Consequently, the BEMS must improve its decisions directly from interaction data. This focus reflects practical deployment conditions in which accurate building models are expensive to develop, environment statistics vary across sites, and controllers must remain reliable under changing operating regimes.

The primary objective is to minimize the cumulative energy bill over a finite control horizon. Cost minimization is interpreted in an operational sense: the BEMS chooses charging/discharging actions that trade off immediate grid exchange costs against future opportunities, considering that storage actions affect SOC and therefore future feasibility.

4.1.1 Exogenous Time Series and Files

This experiment uses a beta version, "v2.2b0," of CityLearn, an open-source Farama Gymnasium environment for single- and multi-agent RL in building/district energy coordination, to benchmark demand response controllers that act on DERs [53].

With the CiltyLearn default datasets, we have ready-to-run scenarios for single-family homes with PV/batteries and commercial buildings, and 1 year of hourly signals, including building electricity data alongside weather, electricity prices, and carbon-intensity time series, as listed in Table 4.1.

File	**Key columns**	**Units**
Building data	`month`, `hour`, `day_type`, `non_shiftable_load`, `solar_generation`, ...	`hour/day_type/month`: index; loads/demands: kWh; `solar_generation`: W/kW
Weather data	`diffuse_solar_irradiance`, `direct_solar_irradiance`, and 6h/12h/24h forecasts	W/m^2
Carbon intensity	`carbon_intensity`	kgCO$_2$/kWh
Pricing data	`electricity_pricing`, `electricity_pricing_predicted_6h`, `..._12h`, `..._24h`	$/kWh
Schema	device definitions, observation/action defaults, `seconds_per_time_step`	n/a

Table 4.1 Key Files and Mappings in CityLearn Dataset

4.1.2 Electrical Energy Storage Unit (Battery) Model

The section describes the mathematical model for the electrical ESU used in the environment. Table 4.2 introduces the variables used for modelling.

Table 4.2 Parameters and variables used for the battery model.

Symbol	Definition	Unit
t	Discrete time-step index	—
a_t	Agent action at time t (normalized charge/discharge request; $+$ charge, $-$ discharge)	—
C	Maximum (nameplate) energy capacity of the battery	kWh
P_{nom}	Nominal power limit (nameplate)	kW
P_{avail}	Available nominal power limit	kW
$P_{\max}^{\text{in}}$	Max admissible charging power at current SOC from the capacity–power curve (charge limit)	kW
$P_{\max}^{\text{out}}$	Max admissible discharging power at current SOC from the capacity–power curve (discharge limit)	kW
λ	Loss coefficient (fractional energy loss per time step)	—
η	Base technical round-trip efficiency	—
κ	Capacity loss / degradation coefficient	—
DoD	Depth of Discharge (maximum discharge fraction)	—
s_t	State of charge at time t	—
s_{t-1}	State of charge at previous time step	—
E_t^{req}	Requested energy for step t after mapping action to energy	kWh
E_t^{dod}	DoD-based discharge-limit energy	kWh
E_t^{init}	Battery energy at the start of step t	kWh
E_t	Battery stored energy at the end of step t after applying (dis)charge and saturation limits	kWh
E_t^{bat}	Energy at battery terminals	kWh
$C_{\text{deg},t}$	Available degraded capacity at time t	kWh

The following set of equations describes the battery dynamics:

a. **Action to requested energy:**

$$E_t^{\text{req}} = a_t\, C \tag{11}$$

b. **Apply limits (charge/discharge):**

$$\begin{aligned} E_t^{\text{req}} &\leftarrow \min\left(E_t^{\text{req}},\ P_{\max}^{\text{in}},\ P_{\text{avail}},\ C_{\text{deg}} - E_t^{\text{init}}\right) && \text{if } E_t^{\text{req}} \geq 0 \\ E_t^{\text{req}} &\leftarrow \max\left(E_t^{\text{req}},\ -P_{\max}^{\text{out}},\ E_t^{\text{dod}}\right) && \text{if } E_t^{\text{req}} < 0 \end{aligned} \tag{12}$$

$$E_t^{\text{dod}} = -\max\left((s_{t-1} - (1 - \text{DoD}))\, C\, \sqrt{\eta},\ 0\right). \tag{13}$$

c. **Energy after standby losses:**

$$E_t^{\text{init}} = \max\left(0,\ s_{t-1}\,C\,(1-\lambda)\right) \tag{14}$$

d. **SOC update:**

$$E_t = \begin{cases} \min\left(E_t^{\text{init}} + E_t^{\text{req}}\sqrt{\eta},\ C\right), & E_t^{\text{req}} \geq 0 \\ \max\left(0,\ E_t^{\text{init}} + \dfrac{E_t^{\text{req}}}{\sqrt{\eta}}\right), & E_t^{\text{req}} < 0 \end{cases} \tag{15}$$

$$s_t = \frac{E_t}{C}. \tag{16}$$

e. **Energy balance used in electricity accounting:**

$$E_t^{\text{bat}} = \begin{cases} \dfrac{E_t - E_t^{\text{init}}}{\sqrt{\eta}}, & E_t - E_t^{\text{init}} \geq 0 \\ (E_t - E_t^{\text{init}})\sqrt{\eta}, & E_t - E_t^{\text{init}} < 0 \end{cases} \tag{17}$$

f. **Capacity degradation:**

$$C_{\text{deg},t} = \max\left(C_{\text{deg},t-1} - \frac{\kappa\,C\,|E_t^{\text{bat}}|}{2\,C_{\text{deg},t-1}},\ 0\right) \tag{18}$$

4.1.3 Solar PV Model

The section describes the mathematical model for the PV used in the environment. Table 4.3 introduces the variables used for modelling.

Table 4.3 Parameters/Variables used for PV Modeling.

Symbol	**Definition**	**Unit**
t	Discrete time-step index	—
P_{pv}	PV system nominal power (nameplate)	kW
p_t^{inv}	Inverter AC output per kW	W/kW
G_t	PV generated energy	kWh
E_t^{pv}	PV generation energy represented as negative load	kWh

Equation 19 shows the calculation of PV-generated energy at each timestep t

$$G_t = \frac{P_{\text{pv}}\,p_t^{\text{inv}}}{1000} \tag{19}$$

This PV generation is stored as a negative load

$$E_t^{\text{pv}} = -G_t \tag{20}$$

Then the observation "`solar_generation`" is reported as E_t^{pv}.

4.1.4 Demand Model

In BEMS, electrical loads are commonly classified into several categories, including non-shiftable, shiftable & non-interruptible, and controllable loads. Non-shiftable loads E_t^{nsl} (e.g., televisions and microwave ovens) require their power demand to be met immediately and in full, without scheduling flexibility. Shiftable & non-interruptible loads E_t^{nil} (e.g., washing machines) can be deferred to an appropriate time; however, once initiated, their operation cannot be interrupted. By contrast, controllable loads E_t^{cl} (e.g., HVAC systems, heat pumps and electric water heaters) can be regulated more continuously, allowing their operating times and energy consumption to be adjusted within predefined constraints (e.g., maintaining indoor temperature within an acceptable range).

In this thesis, all loads are modeled as non-shiftable. Accordingly, the agent's control authority is restricted to the electrical ESU, represented here by a battery pack. This simplified formulation serves as an initial step for training and evaluating the Meta-RL agent under previously unseen operating conditions. Building on this baseline, subsequent work will extend the action space to incorporate shiftable and controllable loads, enabling more of the demand-side flexibility.

$$E_t^{\mathrm{demand}} = E_t^{\mathrm{nsl}} + E_t^{\mathrm{nil}} + E_t^{\mathrm{cl}} \tag{21}$$

4.1.5 Energy Balance / Grid Interaction

Net electricity consumption:

$$E_t^{\mathrm{net}} = E_t^{\mathrm{demand}} + E_t^{\mathrm{bat}} + E_t^{\mathrm{pv}} \tag{22}$$

where E_t^{pv} is negative (generation) and E_t^{bat} is positive for charging and negative for discharging.

Grid import/export:

$$E_t^{\mathrm{import}} = \max(0, E_t^{\mathrm{net}}), \quad E_t^{\mathrm{export}} = \max(0, -E_t^{\mathrm{net}}) \tag{23}$$

Cost and emissions:

$$\mathrm{cost}_t = E_t^{\mathrm{net}} \cdot \pi_t, \quad \mathrm{emission}_t = \max(0, E_t^{\mathrm{net}} \cdot c_t) \tag{24}$$

with price π_t and carbon intensity c_t.

4.2 Markov Decision Process Formulation

4.2.1 Problem Setup

We consider N buildings $B = \{B_1, \ldots, B_N\}$. For each building B_i, The BEMS problem addressed in this work aims to minimise the total operating cost of electricity consumption. This optimisation task can be formulated as a MDP as follows:

$$M_i = \langle S_i, A_i, R_i, \gamma_i, P_i \rangle.$$

- S_i denotes the set of states describing the building i at a given time step,
- A_i represents the set of control actions available to the agent,
- R_i is the immediate reward obtained after taking an action in a given state,
- P_i is the state transition probability distribution that governs the system evolution from one state to the next under a chosen action, and
- $\gamma_i \in [0, 1]$ is the discount factor that determines the relative importance of future rewards. Values of γ close to 1 emphasise long-term performance, whereas smaller values prioritise immediate outcomes.

At time t, the agent observes a state, chooses action a_t, transitions to the next state, and receives reward r_{t+1} as illustrated in Figure 4.1. Given episode length H_i, the goal is to learn a policy $\pi_i(a \mid s)$ which then translates into a state-value function Equation 31 and an action-value function Equation 32 which represents the discounted sum of rewards.

Within the standard MDP formulation, the agent is assumed to have complete and accurate access to the environment state at every time step. In practical energy management settings, however, this assumption is often violated. Not all state variables are directly and accurately measurable or observable. To account for such limitations, the MDP is extended to a Partially Observable Markov Decision Process (POMDP) [34][5].

A POMDP augments the MDP with an observation model and is defined by:

$$M_i = \langle S_i, A_i, R_i, \gamma_i, P_i, O_i, Z_i \rangle.$$

- O_i is the observation space, representing the measurable signals available to the agent (i.e. an observable subset of the full set of states S_i), and
- Z_i is the observation probability function, which specifies the likelihood of receiving a particular observation given the underlying state and the executed action.

In the environment considered here, transitions depend on hidden exogenous sequences and device states, including the full time index, exogenous time series, and device states/parameters. Therefore, the agent must rely on history to infer hidden variables, making the problem partially observable.

This extension enables the agent to operate in settings where the complete system state is not directly accessible. In this context, S_i can be viewed as the superset of all latent system states, whereas O_i represents the subset of S_i that is actually observable. Consequently, the agent must infer or estimate the unobserved components of the state by combining available measurements with an implicit or explicit probabilistic model, often leveraging information from past observations [22].

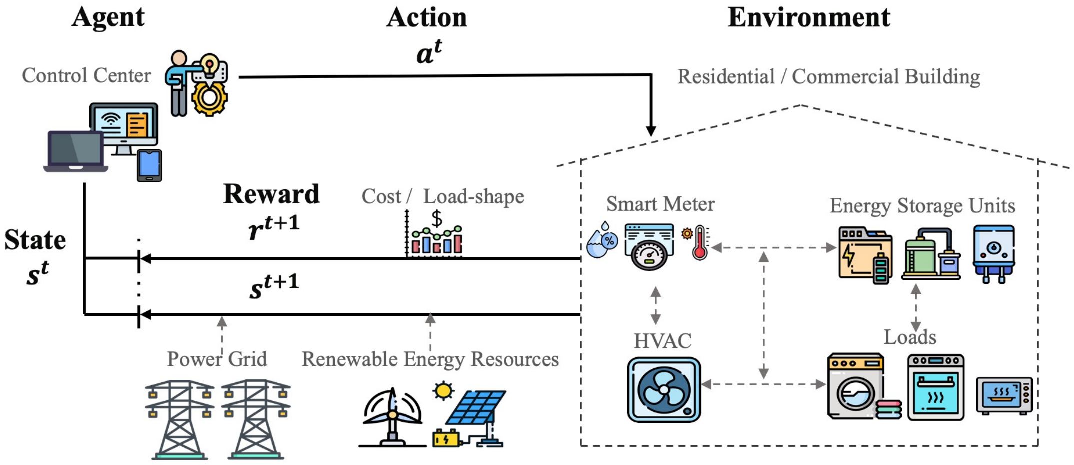


Figure 4.1 A general diagram for RL with key components in BEMS [56].

4.2.2 Observation Model

A. Active Observations Space

Notation: time step t is hourly, $\Delta t = 1\ \mathrm{h}$. Raw values are from data files or internal model state. Normalized values are produced by an observation normalization helper function via:

a. periodic normalization for `hour`, `day_type`, `month`:

$$x_{\sin} = \sin\left(\frac{2\pi x}{x_{\max}}\right), \quad x_{\cos} = \cos\left(\frac{2\pi x}{x_{\max}}\right) \tag{25}$$

b. min-max normalization for all observations:

$$\tilde{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{26}$$

Active observations and computations:

Observation	Definition	Unit	Normalized output
`hour`	Hour-of-day (1–24)	hour index	periodic to `hour_sin`, `hour_cos`, then min-max
`day_type`	Day type (1–8, Mon–Sun + holiday)	index	periodic to `day_type_sin`, `day_type_cos`, then min-max
`solar_generation`	PV generation magnitude	kWh	min-max
`electricity_pricing`	Electricity price	$/kWh	min-max
`electricity_pricing_predicted_6h`	6h-ahead electricity price	$/kWh	min-max
`net_electricity_consumption`	Net building electricity consumption (imports positive, exports negative)	kWh	min-max (bounds estimated)
`electrical_storage_soc`	Battery state of charge	fraction [0,1]	min-max (0–1)

Table 4.4 Observation space of the Energy Management Problem

The experiment activates 7 observations listed in Table 4.4, but the observation wrapper helper function expands periodic variables into sine/cosine pairs and then min-max normalizes all values to $[0, 1]$. The agent, therefore, receives 9 features in this order:

`day_type_cos`, `day_type_sin`, `hour_cos`, `hour_sin`, `solar_generation`, `electrical_storage_soc`, `net_electricity_consumption`, `electricity_pricing`, `electricity_pricing_predicted_6h`, `electricity_pricing_predicted_12h`.

B. Observation Probability Function

Observations are deterministic functions of the internal state, followed by normalization:

$$o_t = g(s_t), \tag{27}$$

where g includes the extraction of each building's active observations. Thus, the observation probability is a Dirac delta:

$$Z(o_t \mid s_t, a_{t-1}) = \begin{cases} 1, & \text{if } o_t = g(s_t), \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$

4.2.3 Action Space

Action (single building): one continuous action for electrical ESU (battery).

- **Action name:** `electrical_storage`
- **Meaning:** fraction of battery capacity to charge/discharge per step.
- **Sign convention:** positive = charge, negative = discharge.
- **Bounds:**

$$a_t \in \left[-\min\left(1, \frac{P_{\text{nom}}}{C}\right),\ \min\left(1, \frac{P_{\text{nom}}}{C}\right) \right] \tag{29}$$

4.2.4 Reward Function

The reward function in this experiment computes the total electricity cost using raw (unnormalized) signals. Let p_t^i be electricity pricing and $E_t^{\text{net,i}}$ be net electricity consumption for building i at time t.

$$r_t = -\sum_{i=1}^{N} p_t^i \, E_t^{\text{net,i}}. \tag{30}$$

4.2.5 Value Functions

State-value function:

$$V^\pi(s_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{H-1-t} \gamma^k \, r_{t+k+1} \,\middle|\, s_t \right]. \tag{31}$$

Action-value function:

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{k=0}^{H-1-t} \gamma^k \, r_{t+k+1} \,\middle|\, s_t, a_t \right], \tag{32}$$

with $\gamma = 0.99$ in this experiment.

4.2.6 Base learner Function and Learnable Parameter

Then, the base learner f is defined with learnable parameter θ to map states S_i (or observations O_i) to actions A_i. The effectiveness of function f with optimal parameters θ_i is defined as

$$\mathcal{L}(f_{\theta_i}) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}_i} \left[\left(r + \gamma \max_{a'} Q\Big(s', a'; f_{\theta_i^-}\Big) - Q(s, a; f_{\theta_i}) \right)^2 \right]. \tag{33}$$

where θ_i^- are the parameters of target critic network.

Chapter 5

Methodology

5.1 Proposed Algorithmic Framework

In this section, we present a common meta-learning framework used by all proposed algorithms and then detail the three Meta-RL variants considered. At the beginning of each iteration, a batch of buildings/tasks is sampled from a building/task distribution. The setup follows a two-level hierarchy: a meta-level coordinator learns knowledge that transfers across buildings, while each building is treated as a separate task with its own dynamics and reward statistics.

A single meta-learner owns the shared actor–critic backbone and the algorithm-specific meta component (e.g., a latent context encoder, episodic memory, ...), whereas each building maintains its own experience buffers and task-local adaptation state. Importantly, the building-level interaction loop and adaptation procedure are structurally similar across all three methods; the key differences lie at the meta-level (i) how adaptation happens, and (ii) what mechanism carries information across tasks. The following subsections describe each algorithm, highlighting the corresponding meta-level adaptation and transfer mechanism.

A central practical challenge of Meta-RL algorithms is the high computational cost of collecting multiple trajectories for multiple tasks per meta-update, resulting in poor wall-clock efficiency, as indicated in [57]. To make training feasible, this framework uses parallel rollout collection. Multiple independent environment instances are executed concurrently, producing batched observations and rewards suitable for vectorized processing. From a systems perspective, this corresponds to parallel actors collecting experience for a centralized learner, thereby increasing environment steps per second. The parallelization is integrated into the meta-learning pipeline so that both inner-loop sampling and outer-loop updates benefit from concurrency.

5.2 Context / Latent-variable Meta-Reinforcement Learning

From this family of Meta-RL algorithms, we will use the PEARL-style algorithm, formulated as follows [41]:

A single meta-learner owns the shared policy and value function parameters, while each building maintains its own experience buffers and latent context state . Concretely:

- **Shared vs. adapted components**: a single Gaussian actor and twin Q critics (with target networks) are shared globally across buildings. Building-specific adaptation is performed through a latent variable z_i inferred from recent experience; there are no per-building network copies in the implementation.
- **Buffers**: each building has a replay buffer D_i for off-policy SAC updates and an implicit context buffer C_i formed by the most recent transitions sampled from D_i. The context encoder is shared globally.
- **Meta-level aggregation**: all building updates act on shared parameters using shared optimizers, effectively aggregating gradients across tasks into a single global update step.

5.2.1. Notation

Symbol	Meaning
$i \in \{1, \ldots, N\}$	Building (task) index
$\mathcal{T}_i \sim p(\mathcal{T})$	Task/building sampled from task distribution
$s_t^i, a_t^i, r_t^i, s_{t+1}^i, d_t^i$	Transition tuple for building i
D_i	Replay buffer for building i
C_i	Context buffer (recent transitions) for building i
z_i	Latent variable encoding building context
μ_i	Mean-pools across a context batch
σ_i	Standard deviation of the Gaussian posterior
$q_\varphi(z_i \mid C_i)$	Context encoder (amortized inference)
$\pi_\theta(a \mid s, z)$	Gaussian actor (shared)
Q_{ψ_1}, Q_{ψ_2}	Twin critics (shared)
$\bar{Q}_{\psi_1}, \bar{Q}_{\psi_2}$	Target critics
α	SAC temperature (entropy weight)
y_t^i	TD target for building i
$\mathcal{L}_Q^{(i)}, \mathcal{L}_\pi^{(i)}, \mathcal{L}_{enc}^{(i)}, \mathcal{L}_\alpha^{(i)}$	Critic, actor, encoder and temperature losses for building i
γ	Discount factor
τ	Polyak averaging factor for target critics
β	KL coefficient
η	Learning rate(s) for actor/critic/encoder/temperature updates
$\Theta = \{\theta, \psi, \varphi, \alpha\}$	Meta-level parameters
$\mathcal{H}_{\text{target}}$	Target entropy

5.2.2. Building-Level Adaptation

A. Transition collection and buffers

At each time step, the shared policy is conditioned on the building-specific latent z_i. The environment returns a transition $(s_t^i, a_t^i, r_t^i, s_{t+1}^i, d_t^i)$ which is stored in the building replay buffer D_i. The context buffer C_i is implemented by sampling the most recent B_c transitions from D_i (a sliding window); this is used to infer z_i.

B. Latent inference from context

The context encoder consumes concatenated transition features $c_t^i = [s_t^i, a_t^i, r_t^i, s_{t+1}^i, d_t^i]$ and mean-pools across a context batch to produce a Gaussian posterior. If the flag "Use Deterministic z" is enabled, $z_i = \mu_i$; otherwise z_i is sampled.

$$\begin{aligned}(\mu_i, \log \sigma_i^2) &= f_\varphi(C_i),\\ q_\varphi(z_i \mid C_i) &= \mathcal{N}\big(\mu_i, \operatorname{diag}(\sigma_i^2)\big)\,,\\ z_i &= \mu_i + \sigma_i \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).\end{aligned} \tag{34}$$

C. SAC targets and losses conditioned on z_i

Twin critics and the actor take the augmented state $[s_t^i, z_i]$. Actions for targets are sampled from the current policy.

$$\begin{aligned}y_t^i = r_t^i + (1 - d_t^i)\gamma\Big(&\min_{k\in\{1,2\}} \bar{Q}_{\psi_k}(s_{t+1}^i, a_{t+1}^i, z_i)\\ &- \alpha \log \pi_\theta(a_{t+1}^i \mid s_{t+1}^i, z_i)\Big).\end{aligned} \tag{35}$$

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,r,s',d)\sim D_i}\Big[(Q_{\psi_1}(s,a,z_i) - y_t^i)^2 + (Q_{\psi_2}(s,a,z_i) - y_t^i)^2\Big] + \beta \mathcal{L}_{enc}. \tag{36}$$

$$\mathcal{L}_\pi = \mathbb{E}_{s\sim D_i,\, a\sim\pi_\theta}\Big[\alpha \log \pi_\theta(a \mid s, z_i) - \min_{k\in\{1,2\}} Q_{\psi_k}(s,a,z_i)\Big]. \tag{37}$$

$$\mathcal{L}_{enc} = \mathrm{KL}(q_\varphi(z_i \mid C_i) \,\|\, \mathcal{N}(0, I))\,. \tag{38}$$

The temperature is updated to track a target entropy $\mathcal{H}_{\text{target}}$.

$$\mathcal{L}_\alpha = -\mathbb{E}_{a\sim\pi_\theta}\Big[\log\alpha\,(\log \pi_\theta(a \mid s, z_i) + \mathcal{H}_{\text{target}})\Big]. \tag{39}$$

D. Inner-loop parameter update

For each building update step, shared parameters are optimized using the building-specific data and inferred z_i. Target critics are updated with Polyak averaging using τ.

$$\begin{aligned}\theta &\leftarrow \theta - \eta_\pi \nabla_\theta \mathcal{L}_\pi,\\ \psi &\leftarrow \psi - \eta_Q \nabla_\psi \mathcal{L}_Q,\\ \varphi &\leftarrow \varphi - \eta_\varphi \nabla_\varphi \mathcal{L}_{enc},\\ \alpha &\leftarrow \alpha - \eta_\alpha \nabla_\alpha \mathcal{L}_\alpha.\end{aligned} \tag{40}$$

5.2.3. Meta-Learning Across Buildings

Buildings are treated as tasks sampled from a distribution $p(\mathcal{T})$. Information transfers across tasks through:

- **Shared initialization and parameters**: $\Theta = \{\theta, \psi, \varphi, \alpha\}$ are global and updated using experience from all buildings.
- **Amortized inference**: the shared encoder $q_\varphi(z_i \mid C_i)$ captures task-specific structure and conditions the policy/value functions.

An outer-loop objective can be written in terms of post-adaptation performance on a query set D_i^q after inferring z_i from a support set C_i^s:

$$\min_{\Theta} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\mathcal{L}^{(i)}_{query}(\Theta; z_i)\right], \quad z_i \sim q_\varphi(z_i \mid C_i^s). \tag{41}$$

$$\Theta \leftarrow \Theta - \beta \nabla_\Theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}^{(i)}_{query}(\Theta; z_i). \tag{42}$$

The meta-update is realized implicitly: the shared optimizers update Θ using aggregated gradients from all building updates, so the global parameters drift toward solutions that generalize across buildings.

The full Meta-learning process is outlined in algorithm 5.1.

5.2.4. Testing and Evaluation

- **Zero-shot behavior**: the evaluation helper function is designed to use a policy adapter with $z_i = 0$ (the prior mean) when no context is provided. This corresponds to empty-context zero-shot execution.
- **Online context inference**: the agent supports periodic context inference; z_i can be updated every step (interval $= 1$) or on a fixed window using recent transitions from C_i and Eq. (34). If the buffer is too small, z_i defaults to zeros.
- **Frozen weights at test time**: actor/critic parameters are not updated during evaluation; only z_i may change if online inference is enabled. Throughout deterministic evaluation and testing, z_i remains fixed at the prior.

The full Meta-testing process is outlined in algorithm 5.2.

Algorithm 5.1: Meta-training loop of Context / Latent-variable meta actor-critic

```
for *episode* $\leftarrow 1$ to $E$ do
    obs $\leftarrow$ ResetAllBuildings$()$
    foreach $i \in \mathcal{B}$ do
        initialize context_state and context buffer $C_i$
    end
    for $t \leftarrow 1$ to $T$ do
        foreach $i \in \mathcal{B}$ do
            $z_i \leftarrow$ InferContext$(C_i)$                      // Equation 34
            sample $a_i \sim \pi_\theta(\cdot \mid s_i, z_i)$
        end
        $(s_{\text{next}}, r, d) \leftarrow$ StepEnvs$(a)$
        foreach $i \in \mathcal{B}$ do
            store $(s_i, a_i, r_i, s_{\text{next},i}, d_i)$ in $D_i$; update $C_i$
        end
        foreach $i \in \mathcal{B}$ do
            $y_i \leftarrow$ BellmanTarget$(\ldots)$                 // Equation 35
            $L_Q^{(i)} \leftarrow$ CriticLoss$(\ldots)$               // Equation 36
            $L_\pi^{(i)} \leftarrow$ ActorLoss$(\ldots)$              // Equation 37
            $L_{\text{enc}}^{(i)} \leftarrow$ EncoderKL$(\ldots)$      // Equation 38
            $L_\alpha^{(i)} \leftarrow$ TemperatureLoss$(\ldots)$     // Equation 39
            update shared params                                    // Equation 40
        end
    end
    compute query loss across buildings                             // Equation 41
    meta-update shared parameters                                   // Equation 42
end
```

Algorithm 5.2: Meta-testing loop of Context / Latent-variable meta actor-critic

```
foreach $i \in \mathcal{B}_{test}$ do
    obs $\leftarrow$ ResetEnv$(i)$
    $z_i \leftarrow 0$                                   // prior mean (zero-shot)
    for $t \leftarrow 1$ to $T$ do
        if $\mathtt{online\_inference}$ and $(t \bmod \mathtt{context\_update\_interval} = 0)$ then
            $z_i \leftarrow$ InferContext$(C_i)$                      // Equation 34
        end
        $a_i \leftarrow \pi_\theta(\cdot \mid s_i, z_i)$
        $(s_{\text{next}}, r, d) \leftarrow$ StepEnv$(a_i)$
        append $(s_i, a_i, r_i, s_{\text{next},i}, d_i)$ to $C_i$
    end
    // actor/critic weights remain frozen (no Equation 40 updates)
end
```

5.3 Memory-based / episodic Meta-Reinforcement Learning

From this family of Meta-RL algorithms, we will use the MANN-style algorithm, formulated as follows [44]:

A single meta-learner owns the shared actor-critic and memory components, while each building maintains its own experience buffers and local history. Concretely:

- **Shared vs. adapted components**: a single Gaussian actor and twin Q critics (with target networks) are shared globally across buildings. Building-specific adaptation is performed through memory retrieval using a global episodic memory; there are no per-building network copies in the implementation.
- **Buffers**: each building has a replay buffer D_i for off-policy SAC updates and a recent history window $H_{i,t}$ used to query memory. The memory encoder is shared globally.
- **Meta-level aggregation**: all building updates act on shared parameters using shared optimizers. The shared components include θ, φ_1, φ_2, $\bar{\varphi}_1, \bar{\varphi}_2$, α (fixed or learned), the memory encoder ω, and the global episodic memory $\mathcal{M}$. Adaptation occurs via the memory readout $m_{i,t}$ (and an optional recurrent hidden state if used).

5.3.1. Notation

Symbol	Meaning
$i \in \{1, \ldots, N_B\}$	Building (task) index
$\mathcal{T}_i \sim p(\mathcal{T})$	Task/building sampled from task distribution
$s_t^i, a_t^i, r_t^i, s_{t+1}^i, d_t^i$	Transition tuple for building i
D_i	Replay buffer for building i
$H_{i,t}$	Recent history window for building i at time t
L_H	History length (number of transitions in $H_{i,t}$)
$\mathcal{M} = \{(K_n, V_n)\}_{n=1}^{N_M}$	Global episodic memory with keys K_n and values V_n
$q_{i,t}, k_{i,t}, v_{i,t}$	Query, key, value from the memory encoder
$e_{i,t,n}$	Attention score between query $q_{i,t}$ and key K_n
$w_{i,t,n}$	Attention weight for memory slot n
$m_{i,t}$	Memory readout used for adaptation
$f_\omega(\cdot)$	Memory encoder with parameters ω
$\pi_\theta(a \mid s, m)$	Gaussian actor (shared)
$Q_{\varphi_1}, Q_{\varphi_2}$	Twin critics (shared)
$\bar{Q}_{\varphi_1}, \bar{Q}_{\varphi_2}$	Target critics
α	SAC temperature (entropy weight)
γ, τ	Discount factor & Polyak averaging factor for target critics
B	Mini-batch size for SAC updates
$\mathcal{H}_{\text{target}}$	Target entropy
η	Learning rate(s) for actor/critic/temperature updates

5.3.2. Building-Level Adaptation

A. Transition collection and buffers

At each time step, the shared policy is conditioned on the building-specific memory readout $m_{i,t}$. The environment returns a transition $(s_t^i, a_t^i, r_t^i, s_{t+1}^i, d_t^i)$ which is stored in the building replay buffer D_i. The history window $H_{i,t}$ is formed by the most recent L_H transitions (a sliding window) and is used to query the global episodic memory. If a recurrent state is used, it is computed from $H_{i,t}$ and concatenated with $m_{i,t}$ when conditioning the policy and critics.

B. Memory encoding and readout

The memory encoder consumes the recent history and produces a query, key, and value. The query is used to read from the global memory, while the key/value can be written back to memory.

$$(q_{i,t}, k_{i,t}, v_{i,t}) = f_\omega(H_{i,t}). \tag{43}$$

The memory read uses a differentiable attention over stored keys.

$$e_{i,t,n} = \frac{q_{i,t}^\top K_n}{\sqrt{d_k}}, \qquad w_{i,t,n} = \frac{\exp(e_{i,t,n})}{\sum_{j=1}^{N_M} \exp(e_{i,t,j})}, \qquad m_{i,t} = \sum_{n=1}^{N_M} w_{i,t,n} V_n. \tag{44}$$

The actor and critics condition on $[s_t^i, m_{i,t}]$, enabling fast, implicit adaptation.

C. SAC targets and losses conditioned on $m_{i,t}$

Twin critics and the actor take the memory-augmented state. Actions for targets are sampled from the current policy using the next-step memory readout $m_{i,t+1}$.

$$\begin{aligned} y_t^i = r_t^i + (1 - d_t^i)\gamma\big(&\min_{k\in\{1,2\}} \bar{Q}_{\varphi_k}(s_{t+1}^i, a_{t+1}^i, m_{i,t+1}) \\ &-\alpha \log \pi_\theta(a_{t+1}^i \mid s_{t+1}^i, m_{i,t+1})\big). \end{aligned} \tag{45}$$

$$\mathcal{L}_Q = \mathbb{E}_{(s,a,r,s',d)\sim D_i} \left[(Q_{\varphi_1}(s,a,m_{i,t}) - y_t^i)^2 + (Q_{\varphi_2}(s,a,m_{i,t}) - y_t^i)^2\right]. \tag{46}$$

$$\mathcal{L}_\pi = \mathbb{E}_{s\sim D_i, a\sim\pi_\theta} \left[\alpha \log \pi_\theta(a \mid s, m_{i,t}) - \min_{k\in\{1,2\}} Q_{\varphi_k}(s,a,m_{i,t})\right]. \tag{47}$$

The temperature is updated to track a target entropy $\mathcal{H}_{\text{target}}$ when α is learned.

$$\mathcal{L}_\alpha = -\mathbb{E}_{a\sim\pi_\theta} \left[\log \alpha \left(\log \pi_\theta(a \mid s, m_{i,t}) + \mathcal{H}_{\text{target}}\right)\right]. \tag{48}$$

D. Inner-loop parameter update

For each building update step, shared parameters are optimized using the building-specific data and memory readout. Gradients flow into θ, φ_1, φ_2, and the memory encoder ω through the differentiable readout $m_{i,t}$; α is updated if learned.

$$\begin{aligned}\theta &\leftarrow \theta - \eta_\pi \nabla_\theta \mathcal{L}_\pi, \\ \varphi_k &\leftarrow \varphi_k - \eta_Q \nabla_{\varphi_k} \mathcal{L}_Q, \\ \omega &\leftarrow \omega - \eta_\omega \nabla_\omega (\mathcal{L}_\pi + \mathcal{L}_Q), \\ \alpha &\leftarrow \alpha - \eta_\alpha \nabla_\alpha \mathcal{L}_\alpha.\end{aligned} \tag{49}$$

Target critics are updated with Polyak averaging using ρ.

$$\bar{\varphi}_k \leftarrow \rho\, \bar{\varphi}_k + (1-\rho)\, \varphi_k. \tag{50}$$

aMemory writes are non-differentiable (append/FIFO), while reads remain differentiable.

5.3.3. Meta-Learning Across Buildings

Buildings are treated as tasks sampled from a distribution $p(\mathcal{T})$. Information transfers across tasks through:

- **Shared initialization and parameters**: $\{\theta, \varphi_1, \varphi_2, \omega, \alpha\}$ are global and updated using experience from all buildings.
- **Global episodic memory**: the shared memory $\mathcal{M}$ stores key/value pairs generated across buildings, enabling cross-task reuse of experience prototypes and fast adaptation via memory retrieval.

A simple write rule appends the current key/value and maintains a fixed capacity N_M using FIFO eviction.

$$\mathcal{M} \leftarrow \mathrm{FIFO}(\mathcal{M} \cup \{(k_{i,t}, v_{i,t})\}). \tag{51}$$

An outer-loop objective can be written as an expectation over tasks using memory-conditioned SAC losses.

$$\min_{\Theta}\ \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\mathcal{L}^{(i)}(\Theta; \mathcal{M}) \right], \quad \Theta = \{\theta, \varphi_1, \varphi_2, \omega, \alpha\}. \tag{52}$$

$$\Theta \leftarrow \Theta - \beta \nabla_\Theta \frac{1}{N_B} \sum_{i=1}^{N_B} \mathcal{L}^{(i)}(\Theta; \mathcal{M}). \tag{53}$$

The meta-update is realized implicitly: the shared optimizers update Θ using aggregated gradients from all building updates, so the global parameters and memory learn to support rapid adaptation through memory queries and readouts.

The full Meta-learning process is outlined in algorithm 5.3.

Algorithm 5.3: Meta-training loop of Memory-based Meta-RL

```
for episode $\leftarrow 1$ to $E$ do
    obs $\leftarrow$ ResetAllBuildings$()$
    foreach $i \in \mathcal{B}$ do
        initialize history $H_i$
    end
    for $t \leftarrow 1$ to $T$ do
        foreach $i \in \mathcal{B}$ do
            $(q_i, k_i, v_i) \leftarrow$ MemoryEncoder$(H_i)$            // Equation 43
            $m_i \leftarrow$ MemoryRead$(q_i, \mathcal{M})$              // Equation 44
            sample $a_i \sim \pi_\theta(\cdot \mid s_i, m_i)$
        end
        $(s_{\text{next}}, r, d) \leftarrow$ StepEnvs$(a)$
        foreach $i \in \mathcal{B}$ do
            store $(s_i, a_i, r_i, s_{\text{next},i}, d_i)$ in $D_i$; update $H_i$
        end
        foreach $i \in \mathcal{B}$ do
            $y_i \leftarrow$ BellmanTarget$(\ldots)$                     // Equation 45
            $L_Q^{(i)} \leftarrow$ CriticLoss$(\ldots)$                  // Equation 46
            $L_\pi^{(i)} \leftarrow$ ActorLoss$(\ldots)$                 // Equation 47
            $L_\alpha^{(i)} \leftarrow$ TemperatureLoss$(\ldots)$        // Equation 48
            update shared parameters                                     // Equation 49
            update target critics                                        // Equation 50
            write $(k_i, v_i)$ to $\mathcal{M}$                          // Equation 51
        end
    end
    compute meta objective across buildings                              // Equation 52
    meta-update shared parameters                                        // Equation 53
end
```

5.3.4. Testing and Evaluation

- **Zero-shot behavior**: evaluation freezes the shared parameters θ, φ, ω, and α. The learned global episodic memory $\mathcal{M}$ from training is used for retrieval, and the history window H_t is initialized empty.

- **Online memory-based adaptation**: as transitions accumulate, the history window H_t is updated, queries are recomputed, and m_t changes online. Adaptation occurs via retrieval alone.

The full Meta-testing process is outlined in algorithm 5.4.

Algorithm 5.4: Meta-testing loop of Memory-based Meta-RL

```
foreach $i \in \mathcal{B}_{test}$ do
    obs $\leftarrow$ ResetEnv($i$) & initialize history $H_i$
    for $t \leftarrow 1$ to $T$ do
        $(q_i, \_, \_) \leftarrow$ MemoryEncoder($H_i$)          // Equation 43
        $m_i \leftarrow$ MemoryRead($q_i, \mathcal{M}$)            // Equation 44
        $a_i \leftarrow \pi_\theta(\cdot \mid s_i, m_i)$
        $(s_{\text{next}}, r, d) \leftarrow$ StepEnv($a_i$)
        append $(s_i, a_i, r_i, s_{\text{next},i}, d_i)$ to $H_i$
    end
    // Actor/critic weights remain frozen (no Equation 49 updates).
end
```

5.4 Black-box / Recurrent Meta-Reinforcement Learning

From this family of Meta-RL algorithms, we will use the RL^2-style algorithm, formulated as follows [11]: A single meta-learner owns the shared actor-critic, while each building maintains its own experience buffer and local trajectories history. Concretely:

- **Shared architecture**: a single recurrent GRU actor and twin recurrent critics are shared across all buildings, along with target critics and the entropy temperature α.
- **Per-building state**: each building keeps its own sequence replay buffer D_i and online recurrent hidden states $h_t^{\pi}, h_t^{Q1}, h_t^{Q2}$.
- **Adaptation carrier**: adaptation is entirely implicit via hidden state evolution; there are no explicit latent variables or per-task gradient updates at test time.

5.4.1. Notation

Symbol	Meaning
$i \in \{1, \dots, N_B\}$, $\mathcal{B}$	Building (task) index and set of training buildings
$\mathcal{T}_i \sim p(\mathcal{T})$	Task/building sampled from task distribution
E	Training episodes per building
T	Transitions per episode (environment steps)
G	Gradient steps per environment step
$s_t^i, a_t^i, r_t^i, d_t^i, s_{t+1}^i$	Transition tuple for building i
$a_{t-1}^i, r_{t-1}^i, d_{t-1}^i$	Previous action/reward/done for augmentation
x_t^i	Augmented actor input $[s_t^i, a_{t-1}^i, r_{t-1}^i, d_{t-1}^i]$
$x_t^{Q,i}$	Augmented critic input $[s_t^i, a_t^i, a_{t-1}^i, r_{t-1}^i, d_{t-1}^i]$
$h_t^{\pi,i}, h_t^{Q1,i}, h_t^{Q2,i}$	Recurrent hidden states for actor and critics
π_θ	Recurrent Gaussian actor (shared)
$Q_{\varphi_1}, Q_{\varphi_2}$	Twin recurrent critics (shared)
$\bar{Q}_{\bar{\varphi}_1}, \bar{Q}_{\bar{\varphi}_2}$	Target critics
y_t^i	TD target for building i
$\mathcal{L}_Q^{(i)}, \mathcal{L}_\pi^{(i)}, \mathcal{L}_\alpha^{(i)}$	Critic, actor, and temperature losses for building i
α $(\log \alpha)$	SAC temperature (trainable log-parameter)
$\mathcal{H}_{\text{target}}$	Target entropy
γ	Discount factor
τ	Polyak averaging factor for target critics
η	Learning rate(s) for actor/critic/temperature updates
β	Meta step size in the outer objective
D_i	Replay buffer for building i (sequence storage)
$L_{\text{seq}}, L_{\text{trunc}}$	Configured sequence length and BPTT truncation length
L_t	Effective sequence length, $L_t = \min(L_{\text{seq}}, L_{\text{trunc}})$
L_b	Burn-in length (excluded from loss)
B	Mini-batch size (number of sequences)
$M_{\min}$	Minimum buffer size before updates

5.4.2. Building-Level Adaptation with Recurrent SAC

A. Sequence storage and sampling

- Each building i stores transitions in D_i as contiguous episode segments.
- Each entry is augmented with the previous tuple: $(s_t, a_t, r_t, s_{t+1}, d_t, a_{t-1}, r_{t-1}, d_{t-1})$.
- Training samples contiguous sequences of length L_t from D_i; losses can exclude the first L_b steps (burn-in) to stabilize hidden state.

B. Recurrent forward equations

The actor and critics take augmented inputs and update hidden states, which constitute the task-adaptation mechanism.

$$x_t = [s_t, a_{t-1}, r_{t-1}, d_{t-1}]. \tag{54}$$

$$h_t^\pi = \mathrm{GRU}_\pi(h_{t-1}^\pi, x_t), \qquad (\mu_t, \log\sigma_t) \;= f_\theta(h_t^\pi), \qquad a_t \sim \pi_\theta(\cdot \mid s_t, h_t^\pi). \tag{55}$$

$$\begin{aligned} x_t^Q &= [s_t, a_t, a_{t-1}, r_{t-1}, d_{t-1}], \\ h_t^{Qj} &= \mathrm{GRU}_{Qj}(h_{t-1}^{Qj}, x_t^Q), \\ Q_{\varphi_j}(s_t, a_t, h_t^{Qj}) &= g_{\varphi_j}(h_t^{Qj}). \end{aligned} \tag{56}$$

C. Sequence-based SAC objectives

Targets use the target critics and the next-step recurrent inputs. Terminal flags d_t mask bootstrap terms, and burn-in removes the first L_b steps from losses.

$$\begin{aligned} y_t = r_t + (1 - d_t)\gamma\Big(&\min_{j\in\{1,2\}} \bar{Q}_{\varphi_j}(s_{t+1}, a_{t+1}, h_{t+1}^{Qj}) \\ &- \alpha \log \pi_\theta(a_{t+1} \mid s_{t+1}, h_{t+1}^\pi)\Big). \end{aligned} \tag{57}$$

$$\mathcal{L}_Q = \sum_{t=L_b}^{L_t-1} \Big[(Q_{\varphi_1}(s_t, a_t, h_t^{Q1}) - y_t)^2 + (Q_{\varphi_2}(s_t, a_t, h_t^{Q2}) - y_t)^2\Big]. \tag{58}$$

$$\mathcal{L}_\pi = \sum_{t=L_b}^{L_t-1} \left[\alpha \log \pi_\theta(a_t \mid s_t, h_t^\pi) - \min_{j\in\{1,2\}} Q_{\varphi_j}(s_t, a_t, h_t^{Qj})\right]. \tag{59}$$

$$\mathcal{L}_\alpha = -\sum_{t=L_b}^{L_t-1} \log\alpha \,\big(\log \pi_\theta(a_t \mid s_t, h_t^\pi) + \mathcal{H}_{\text{target}}\big). \tag{60}$$

D. Learning and target updates

- Parameters $\theta, \varphi_1, \varphi_2$ and α are updated via truncated Backpropagation Through Time (BPTT) through the sequence.
- Target critics are updated by Polyak averaging.

$$(\theta, \varphi_1, \varphi_2, \alpha) \leftarrow (\theta, \varphi_1, \varphi_2, \alpha) - \eta \nabla \left(\mathcal{L}_Q + \mathcal{L}_\pi + \mathcal{L}_\alpha \right). \tag{61}$$

$$\bar{\varphi}_j \leftarrow (1 - \tau)\bar{\varphi}_j + \tau \varphi_j, \quad j \in \{1, 2\}. \tag{62}$$

5.4.3. Meta-Learning Across Buildings

Buildings are treated as tasks $\mathcal{T}_i$ sampled from a distribution $p(\mathcal{T})$. Information transfers across tasks through:

- Shared recurrent weights $\theta, \varphi_1, \varphi_2$ capture meta-knowledge.
- The GRU hidden-state dynamics implement a learned adaptation rule from experience.

The outer-loop objective can be written in terms of expected sequence losses across tasks, optimized by stochastic gradient descent (SGD) over shared parameters.

$$\min_{\theta, \varphi_1, \varphi_2, \alpha} \; \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\mathcal{L}_Q^{(i)} + \mathcal{L}_\pi^{(i)} + \mathcal{L}_\alpha^{(i)} \right]. \tag{63}$$

$$\Theta \leftarrow \Theta - \beta \nabla_\Theta \frac{1}{N} \sum_{i=1}^{N} \left(\mathcal{L}_Q^{(i)} + \mathcal{L}_\pi^{(i)} + \mathcal{L}_\alpha^{(i)} \right), \quad \Theta = \{\theta, \varphi_1, \varphi_2, \alpha\}. \tag{64}$$

The full Meta-learning process is outlined in algorithm 5.5.

5.4.4. Testing and Evaluation

- **Zero-shot policy**: shared parameters are frozen at meta-trained values; no gradient updates occur at test time.
- **Hidden state initialization**:
  - $h_0^\pi, h_0^{Q1}, h_0^{Q2}$ are initialized to zeros.
  - A learned h_0 is a common extension; if used, it would replace the zero initialization.
- **Online adaptation**: as each new $(s_t, a_{t-1}, r_{t-1}, d_{t-1})$ arrives, GRU hidden states evolve by Eqs. (54)–(56). Adaptation is purely through hidden-state dynamics.

The full Meta-learning process is outlined in algorithm 5.6.

Algorithm 5.5: Meta-training loop of Black-box / Recurrent Meta-RL

```
for episode $\leftarrow 1$ to $E$ do
    obs $\leftarrow$ ResetAllBuildings(); reset recurrent hidden states and close
      open episodes in each $D_i$
    initialize $a_0, r_0, d_0$ for all buildings to $0$
    for $t \leftarrow 1$ to $T$ do
        foreach $i \in \mathcal{B}$ do
            $x_t^i \leftarrow [s_t^i, a_{t-1}^i, r_{t-1}^i, d_{t-1}^i]$                    // Equation 54
            $h_t^{\pi,i}, a_t^i \sim \pi_\theta(\cdot \mid s_t^i, h_t^{\pi,i})$                    // Equation 55
        end
        $(s_{t+1}, r_t, d_t) \leftarrow$ StepEnvs($a_t$)
        foreach $i \in \mathcal{B}$ do
            store $(s_t^i, a_t^i, r_t^i, s_{t+1}^i, d_t^i, a_{t-1}^i, r_{t-1}^i, d_{t-1}^i)$ in $D_i$
        end
        foreach $i \in \mathcal{B}$ do
            for $g \leftarrow 1$ to $G$ do
                sample $B$ contiguous sequences of length $L_t$ from $D_i$ (if
                  $|D_i| \geq \max(B, M_{\min})$); ignore first $L_b$ steps
                $x_t^{Q,i} \leftarrow [s_t^i, a_t^i, a_{t-1}^i, r_{t-1}^i, d_{t-1}^i]$ and compute $Q_{\varphi_j}$   // Equation 56
                $y_t^i \leftarrow$ Target($\cdot$)                                   // Equation 57
                $\mathcal{L}_Q^{(i)} \leftarrow$ CriticLoss($\cdot$)                             // Equation 58
                $\mathcal{L}_\pi^{(i)} \leftarrow$ ActorLoss($\cdot$)                              // Equation 59
                $\mathcal{L}_\alpha^{(i)} \leftarrow$ TemperatureLoss($\cdot, \mathcal{H}_{\text{target}}$)              // Equation 60
                update shared parameters $(\theta, \varphi_1, \varphi_2, \alpha)$          // Equation 61
                update target critics                                  // Equation 62
            end
        end
        compute meta objective across buildings (logging)             // Equation 63
        meta-update shared parameters                                  // Equation 64
    end
    finalize episode sequences in each $D_i$; record per-building episode returns
end
```

Algorithm 5.6: Meta-testing loop of Black-box / Recurrent Meta-RL

```
foreach $i \in test\_buildings$ do
    initialize $h_0^{\pi}, h_0^{Q1}, h_0^{Q2} \leftarrow 0$            // Equation 55-Equation 56
    prev_a, prev_r, prev_d $\leftarrow 0$
    $s \leftarrow$ ResetEnv$(i)$
    for $t \leftarrow 1$ to $T$ do
        $x_t \leftarrow [s_t,$ prev_a, prev_r, prev_d$]$                // Equation 54
        sample $a_t \sim \pi_\theta(\cdot \mid s_t, h_t^{\pi})$          // Equation 55
        $(s_{\text{next}}, r_t, d_t) \leftarrow$ StepEnv$(a_t)$
        prev_a, prev_r, prev_d $\leftarrow a_t, r_t, d_t$
        update hidden states via GRU recurrence
         // Equation 55-Equation 56
    end
end
```

The shared and algorithm-specific hyperparameters used to initialize the three Meta-RL algorithms are listed in Table B.1 and Table B.2 under Appendix B.

Chapter 6

Evaluation Criteria and Experiment Setup

6.1 Diversity and Randomness Strategies

The central goal of this thesis is to evaluate how well control policies generalize across buildings and operating conditions that differ from those encountered during training. Training is therefore intentionally constructed to (i) expose the agent to structural diversity and (ii) introduce randomness that prevents the policy from overfitting to narrow, repeatable patterns at a single building instance.

We address diversity by training on a heterogeneous set of buildings that vary in load magnitudes and usage patterns, thereby inducing variation in underlying equipment specifications and storage configurations. In addition, the selected buildings span multiple climate zones, introducing systematic shifts in both renewable availability (e.g., PV generation profiles) and electricity demand dynamics across seasons.

We complement this with induced randomness during learning by sampling weekly training episodes from randomized start times across the year and by randomizing the initial SOC of the controlled ESU. Together, these mechanisms expose the controller to a broad distribution of seasonal and operational trajectories.

All experiments are conducted in CityLearn [53], an OpenAI Gym-style building energy management simulator that models a portfolio of buildings with pre-computed demand profiles and renewable generation (e.g., solar PV). Within this environment, the models introduced in chapter 4 are used as the system representation, and the meta reinforcement-learning agents defined in chapter 5 are deployed as controllers for the ESU. The agent issues hourly control actions and, in return, receives state observations and reward signals that drive learning and evaluation.

Climate Zones and Weather conditions

The CityLearn environment provided one year of simulation data for commercial building clusters across four climate zones, as well as for a residential building cluster situated in another climate zone. The energy demand for each building has been pre-simulated using EnergyPlus in a different climatic zone of the USA [53].

Table 6.1 Summary of selected CityLearn built-in datasets.

Dataset name (`citylearn_challenge_`)	**Climate zone**	**Climate type**	**Location**	**Buildings type**
`2022_phase_all`	3B	warm-dry	Fontana, CA	Residential
`2020_climate_zone_1`	2A	hot-humid	New Orleans, LA	Commercial
`2020_climate_zone_2`	3A	warm-humid	Atlanta, GA	Commercial
`2020_climate_zone_3`	4A	mixed-humid	Nashville, TN	Commercial
`2020_climate_zone_4`	5A	cool-humid	Chicago, IL	Commercial

The 2022 dataset contains 17 residential homes within a single climate zone, whereas the four 2020 datasets each contain 9 commercial buildings within one climate zone. Taken together, the commercial datasets form a pool of 36 buildings distributed across four distinct climate zones, introducing systematic variation in demand patterns and renewable generation conditions.

Random Seed and Controlled Randomization

The experiment sets a single base seed. The same seed is passed into the main configuration helper and then deterministically offset to create per-environment seeds during training and evaluation. As a result, building selection, episode start randomization, and initial SOC sampling are all reproducible given this seed and the same execution order.

Buildings

The building pool is assembled by loading the selected datasets and filtering to buildings that include an electrical storage device. Candidate building IDs are then ordered consistently (by their numeric suffix) before any selection is performed. The training subset is drawn via seed-controlled sampling, and the test subset is created with a different fixed seed while explicitly excluding the training buildings, ensuring the test set consists entirely of previously unseen buildings. For evaluation, the “Seen” condition reuses the training buildings, whereas the “Unseen” condition evaluates on the held-out test buildings.

Vectorized Environments and Parallel Rollout Collection

Training is run in a parallelized per-building setup: one simulator instance is created for each training building and these instances are bundled into a single vectorized environment with monitoring enabled. As a result, the rollout batch size equals the number of training buildings. To keep per-building randomness independent while remaining reproducible, each building instance is assigned a distinct seed derived from the common base seed plus a fixed per-building offset.

Rollout collection is synchronous across buildings. Each episode starts by resetting all building simulators at once, after which the system advances in lockstep for a fixed number of hourly transitions. At every time step, the controller outputs a batch of actions (one per building), the vectorized environment returns batched next observations and rewards, and the resulting transitions are stored in batch form for learning.

Note that this vectorization is implemented as batched stepping in a single process (rather than multiprocessing), but it still produces simultaneous trajectories for all training buildings within each episode.

Random Initial State of Charge

To diversify initial conditions, the setup randomizes the battery's starting SOC at the beginning of each episode. Concretely, at every environment reset, a new SOC is drawn uniformly from a specified interval and injected as the storage SOC for the first time step. Because the underlying random number generator is seed-controlled, the resulting SOC draws are reproducible given the episode/reset order. This mechanism adds controlled stochasticity to the episode initialization while leaving the within-episode physics and transition dynamics unchanged.

During training, SOC randomization is enabled with bounds spanning the full feasible range, so each building starts every training episode from a uniformly sampled SOC between 0 and 1. In contrast, evaluation and testing disable SOC randomization for both "Seen" and "Unseen" conditions, meaning evaluation rollouts rely on the simulator's default SOC initialization for benchmarking.

Episodes

To increase seasonal coverage without requiring extremely long rollouts, this work adopts short-horizon episodes, one-week segments randomly sampled across the year for each building/task. Episodes are 168 hourly steps (24×7) sampled from the full 8760-hour year. Randomized weekly start times expose the controller to diverse seasonal and operating conditions (e.g., winter vs summer & weekend vs weekday). Short episodes also reduce variance in long-horizon return estimation and allow more frequent learning updates.

Training uses a fixed number of rollouts per building. Each building is trained for ten one-week episodes in a parallel setup with one environment instance per building. In contrast, evaluation and testing are performed with a single rollout per building under a deterministic protocol. Periodic evaluation switches off episode-start randomization and runs one fixed-start episode per building using a base seed with a consistent per-building offset. The final test phase follows the same approach, again disabling episode randomization and executing exactly one deterministic rollout per test building with a seed schedule that shifts the base seed by a constant offset. This procedure makes evaluation reproducible and seed-controlled.

6.2 Evaluation Criteria

6.2.1 Control Methods for Comparison

To evaluate the effectiveness of our 3 Meta-RL algorithms, we compare them with several representative methods described as follows.

A. No Control

A no-control scenario (i.e., no-load shifting) is considered as the first benchmark.

B. Rule-Based Controller

A RBC defined by CityLearn, making decisions based only on the hour of the day, and uses the following action-time-map, and is considered as the second benchmark.

Table 6.2 Action schedule time map for Rule-Based Controller used as benchmark.

Time interval	Action
07:00 – 15:00	Discharge -2.0%
16:00 – 18:00	Discharge -4.4%
19:00 – 22:00	Discharge -2.4%
23:00 – 00:00	Charge $+3.4\%$
00:00 – 06:00	Charge $+5.532\%$

Positive values indicate charging;
negative values indicate discharging.
Rates are expressed as % of maximum capacity per hour.

C. Pretrained Soft Actor-Critic Agent

A SAC agent pretrained on the same experimental setup demonstrated in section 6.1 and uses an average of trained models' parameters as initialization for a new building during testing and evaluation. It's effectively multi-task training (shared policy across buildings) but not meta-learning. [21]

6.2.2 Key Performance Indicators

Table 6.3 Key Performance Indicators (KPIs) and Definitions

KPI	Unit	Definition
Imports	kWh	Net grid imported electricity over the evaluation period.
Exports	kWh	Net exported electricity to the grid over the evaluation period.
Net Electricity Consumption	kWh	Imported minus exported electricity.
Cost	$	Total cost of net imported electricity.
Revenue	$	Total revenue from net exported electricity.
Electricity Bill	$	Total cost minus total revenue.
Daily Peak Average	kW	Average daily peak demand over the evaluation period.
Ramping	kWh	Average electricity demand change from one timestep to the next.
1-load factor	–	Average electricity demand divided by its maximum peak.
Carbon Emissions	$kgCO_2$	Total generated carbon emissions from grid imports.

For comparison convenience, all scores are normalized to the benchmark RBC score. A score of 0.96 means that the RL controller performed 4% better than the baseline RBC.

6.3 Experiment Setup

Building on the discussion in chapters 2 and 3, the key advantage of meta reinforcement learning over standard reinforcement learning is its ability to adapt (i.e., to “learn to learn”) when deployed in new environments that may exhibit a substantial distribution shift relative to the training set. On the other hand, the principal deployment claim within the scope of this thesis is strong out-of-the-box performance under distribution shift, which mimics practical constraints that may limit online learning in operational buildings.

In order to evaluate generalization, the following experimental setup was adopted:

1. **Building split:** For each run, we randomly sample a batch of four buildings for meta-training and a disjoint set of four buildings for held-out assessment.

2. **Meta-Training:** Training is run in a parallelized per-building setup, and a meta-level coordinator learns transferable knowledge across buildings.

3. **Evaluation (seen):** After training and before testing, the meta-learned agent is executed once using a deterministic rollout protocol to record performance KPIs on the training (seen) buildings.
4. **Testing (unseen, zero-shot):** The trained agent is then deployed on previously unseen buildings at randomly selected time periods without any further parameter updates. Performance is assessed immediately in terms of the energy bill and the same KPI suite.

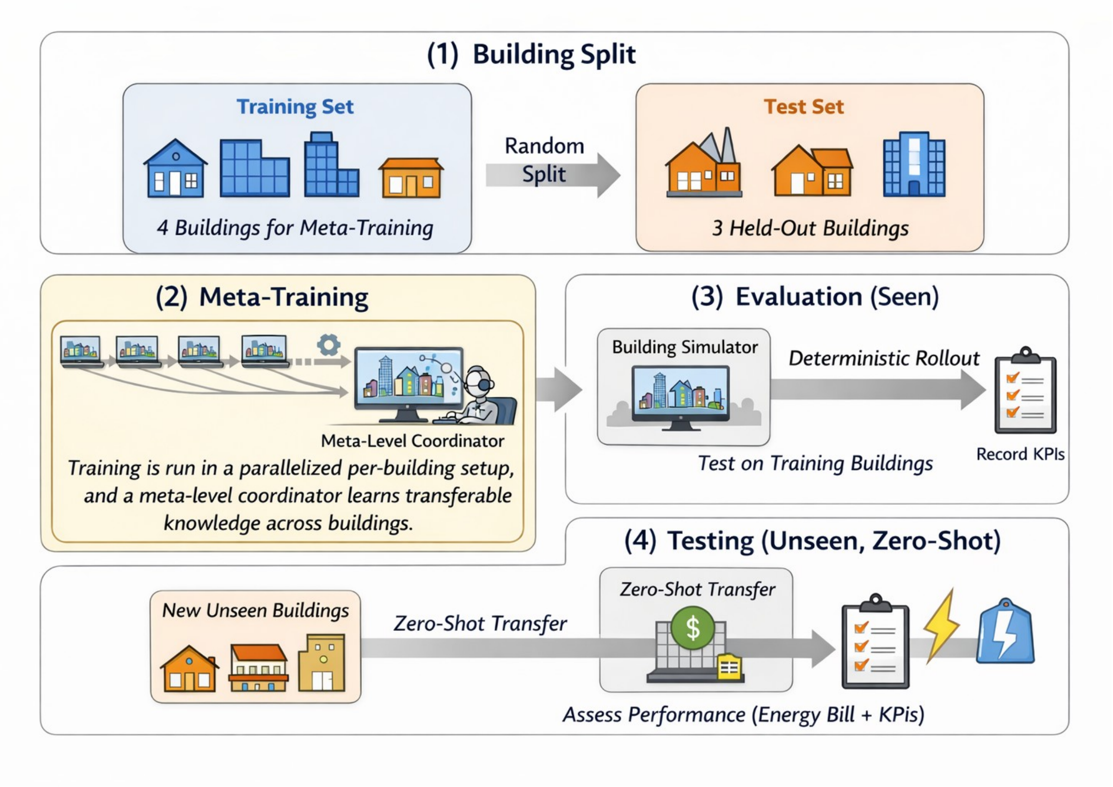

(1) Building Split
Training Set
4 Buildings for Meta-Training
Random Split
Test Set
3 Held-Out Buildings
(2) Meta-Training
Meta-Level Coordinator
Training is run in a parallelized per-building setup, and a meta-level coordinator learns transferable knowledge across buildings.
(3) Evaluation (Seen)
Building Simulator
Deterministic Rollout
Test on Training Buildings
Record KPIs
(4) Testing (Unseen, Zero-Shot)
New Unseen Buildings
Zero-Shot Transfer
Zero-Shot Transfer
Assess Performance (Energy Bill + KPIs)

Figure 6.1 End-to-End Evaluation Protocol for Meta-Learned Controllers

We define two experimental domains, *Residential* and *Commercial*, and repeat each experiment under three different random seeds to quantify variability due to stochastic training and sampling. Across both domains, we evaluate the three Meta-RL algorithms introduced in chapter 5.

Chapter 7

Results

7.1 Limited Cross-Type Generalization

When trained jointly on a mixed set of residential and commercial buildings (sampled from the combined pool of buildings drawn from the five datasets defined in chapter 6), the Meta-RL approach did not generalize effectively across building types. In particular, a single meta-trained policy failed to achieve reliable control performance for both residential and commercial buildings simultaneously.

Since the reward function is defined as (price × net electricity consumption), a primary cause is a reward-scale mismatch induced by strong differences in demand magnitude between building categories. Commercial buildings exhibit electricity demand approximately 10–20× higher than residential buildings, leading to reward signals that are not comparable in scale even when the optimization objective is identical (i.e., cost minimization). Consequently:

- The magnitude and variance of reward are significantly larger for commercial buildings.
- The inner-loop learning signal for residential buildings becomes relatively small and/or noisier compared to commercial buildings.
- Meta-RL tries to learn how to learn, but if the inner-loop learning signal is much noisier/smaller for residential buildings, it adapts poorly or not at all.

To conclude, Joint training across residential and commercial buildings leads to poor residential performance due to reward-scale mismatch. In this setting, Meta-RL exhibits strong sensitivity to reward magnitude, causing commercial-building tasks to dominate the meta-learning updates.

To address this issue, the experiment was separated into two independent Meta-RL runs:

1. **Residential training:** 17 buildings within a single climate zone.
2. **Commercial training:** 36 buildings spanning four climate zones.

This separation reduces cross-type scale imbalance and yields more consistent learning signals within each task family.

7.2 From Raw Runs to KPIs

Tables Table 7.1 and Table 7.2 summarize the performance of the trained Meta-RL agents and the pretrained SAC baseline. Table 7.1 reports the residential results, whereas Table 7.2 reports the commercial results.

Each experiment was repeated with three random seeds. For each seed, four buildings were assigned to the meta-training and evaluation set (“Eval”), and a disjoint set of four buildings was reserved for testing (“Test”). A complete run was executed for each seed, and KPIs were normalized with respect to the RBC baseline as

$$\eta = \left(\frac{X_{\text{agent}}}{X_{\text{RBC}}} \right), \tag{65}$$

where X_{agent} denotes the KPI attained by the RL agent and X_{RBC} denotes the KPI attained by RBC under identical conditions.

For each seed, η was averaged across the four Eval buildings and, separately, across the four Test buildings to obtain seed-level mean values. The final reported mean values μ correspond to the mean of these seed-level values across the three seeds.

Table 7.3 reports the mean μ and standard deviation σ of the Net Electricity Bill KPI for three Meta-RL algorithms MANN, RL^2, PEARL) and a pretrained deep RL baseline SAC, normalized to a RBC set to 100 (lower values indicate lower bills and thus better performance). Results are averaged over three random seeds and are shown separately for an "Eval" split (seen during meta-training) and a disjoint "Test" split (unseen buildings), under both residential and commercial settings.

KPI	Scenario	Pretrained SAC	Memory-based / episodic Meta-RL	Black-box / Recurrent Meta-RL	Context / Latent -variable Meta-RL
Imports	Eval	1.054	1.086	1.177	1.191
	Test	0.998	1.072	1.166	1.152
Exports	Eval	4.918	4.721	7.268	7.126
	Test	1.725	1.708	2.495	2.700
Net Electricity Consumption	Eval	1.007	0.977	1.015	1.017
	Test	0.999	0.982	1.022	1.016
Cost of Imports	Eval	0.946	1.029	1.063	1.073
	Test	0.919	1.037	1.075	1.069
Revenue from Exports	Eval	9.722	5.764	10.723	12.115
	Test	2.711	2.414	3.347	3.973
Net Electricity Bill	Eval	0.642	0.699	0.720	0.630
	Test	0.632	0.744	0.737	0.628
Daily Peak Average	Eval	1.198	1.155	1.526	1.484
	Test	1.049	1.083	1.501	1.441
Ramping	Eval	1.345	1.683	1.965	2.106
	Test	1.269	1.624	1.985	1.988
1 – Load Factor	Eval	1.037	1.017	1.067	1.068
	Test	1.024	1.010	1.042	1.051
Carbon Emissions	Eval	1.088	1.106	1.165	1.196
	Test	1.032	1.099	1.156	1.163

Table 7.1 Performance of the Different Meta-RL Techniques in the Residential BEMS Domain

KPI	Scenario	Pretrained SAC	Memory-based / episodic Meta-RL	Black-box / Recurrent Meta-RL	Context / Latent -variable Meta-RL
Imports	Eval	1.122	0.966	1.038	0.999
	Test	1.265	0.977	1.042	1.080
Exports	Eval	3.201	1.882	2.125	1.972
	Test	3.156	1.225	1.696	2.607
Net Electricity Consumption	Eval	1.012	1.019	1.004	1.018
	Test	1.026	0.936	1.001	0.974
Cost of Imports	Eval	0.962	0.947	1.037	0.934
	Test	1.029	0.934	1.059	0.971
Revenue from Exports	Eval	5.467	2.665	2.462	3.175
	Test	5.815	2.061	2.120	4.435
Net Electricity Bill	Eval	0.503	0.734	1.022	0.642
	Test	0.616	0.832	1.024	0.744
Daily Peak Average	Eval	1.595	1.169	1.350	1.220
	Test	2.029	1.352	1.421	1.689
Ramping	Eval	2.194	1.308	1.673	1.506
	Test	3.276	1.599	1.951	2.119
1 – Load Factor	Eval	1.075	1.071	1.093	1.064
	Test	1.205	1.167	1.178	1.204
Carbon Emissions	Eval	1.120	0.972	1.043	1.003
	Test	1.262	0.986	1.051	1.088

Table 7.2 Performance of the Different Meta-RL Techniques in the Commercial BEMS Domain.

Algorithm	**Residential**		**Commercial**	
	Eval **(%)** $\mu \pm \sigma$	**Test** **(%)** $\mu \pm \sigma$	**Eval** **(%)** $\mu \pm \sigma$	**Test** **(%)** $\mu \pm \sigma$
RBC	**100**	**100**	**100**	**100**
Blackbox Meta-RL (RL^2)	72.0 ± 12.9	73.7 ± 20.7	102.2 ± 15.5	102.4 ± 7.4
Memory-based Meta-RL (MANN)	69.9 ± 15.0	74.4 ± 15.8	73.4 ± 42.9	83.2 ± 13.9
Pretrained SAC	64.2 ± 16.2	63.2 ± 24.9	50.3 ± 67.6	61.6 ± 34.0
Context-based Meta-RL (PEARL)	63.0 ± 17.2	62.8 ± 27.5	64.2 ± 52.6	74.4 ± 21.3

Table 7.3 The mean value μ and standard deviation σ of the Net Electricity Bill KPI for the three Meta-RL and the pretrained SAC algorithms.

7.3 Beyond Numerics: Interpretation, Breakdown Points, and Limits

A. Setting-Wise Performance:

I. Residential Setting

In the residential setting, all learning-based methods outperform RBC by a wide margin on both splits. On the *test* split, PEARL and SAC achieve the lowest mean KPIs and are nearly identical: PEARL reaches 62.8 ± 27.5, and SAC reaches 63.2 ± 24.9. MANN and RL^2 perform worse on average but still substantially below the RBC reference, with test results of 74.4 ± 15.8 and 73.7 ± 20.7, respectively. The ranking by test mean is therefore:

$$\text{PEARL} \approx \text{SAC} \; < \; \text{RL}^2 \approx \text{MANN} \; \ll \; \text{RBC}.$$

Across evaluation and test, the mean changes are small for PEARL ($63.0 \rightarrow 62.8$) and SAC ($64.2 \rightarrow 63.2$), indicating limited degradation on unseen residential buildings.

II. Commercial Setting

In the commercial setting, method ranking changes more clearly. On the *test* split, SAC achieves the best mean performance (61.6 ± 34.0), followed by PEARL (74.4 ± 21.3) and MANN (83.2 ± 13.9). RL^2 performs worst and slightly underperforms RBC, with 102.4 ± 7.4. The ranking by test mean is therefore:

$$\text{SAC} \; < \; \text{PEARL} \; < \; \text{MANN} \; \ll \; \text{RBC} \lesssim \text{RL}^2.$$

A second empirical fact is the pronounced variability in the commercial case. Standard deviations are large for SAC and the Meta-RL methods, particularly on the evaluation split (e.g., SAC: 50.3 ± 67.6; PEARL: 64.2 ± 52.6; MANN: 73.4 ± 42.9), indicating strong sensitivity to the sampled buildings and their operational regimes.

B. Residential vs. Commercial Performance Gap.

The residential setting exhibits a smaller spread in test performance and a near tie between PEARL and SAC. This is consistent with a comparatively smoother task family, where unseen buildings remain close to the meta-training distribution and a single well-trained policy can transfer effectively.

For commercial buildings, larger performance gaps between "Eval" & "Test" and larger variances indicate that the commercial setting contains qualitatively different operating modes for some buildings and seeds. In section 7.4, we examine this heterogeneity in operational characteristics and discuss how outlier operating regimes affect the performance of both SAC and context-based Meta-RL.

Further plots highlighting performance difference between residential and commercial setups are shown in Figure C.1 - Figure C.4 under Appendix C.

C. Why soft-actor-critic outperforms Meta-Reinforcement Learning on Commercial Buildings.

The commercial results suggest that the benefits of meta-learning are not consistently realized under strong distribution shift. A plausible explanation is that Meta-RL methods rely on accurate task inference (via memory or latent context) from limited interaction, and mismatches between meta-training and test task families can lead to incorrect or slow identification of the underlying building regime. In cost-minimization problems, early mistakes are directly penalized in the cumulative bill, making inference errors particularly costly. Additionally, the meta-training regime uses a small number of buildings per run, which may be insufficient to learn an adaptation mechanism that spans the multi-modal variability of commercial buildings (e.g., distinct operating regimes and parameter combinations). Under such conditions, a single robust policy trained by SAC (supported by entropy-regularized optimization) may generalize better by avoiding brittle specialization and reducing reliance on explicit task inference.

This interpretation is further supported by the learning-curve evidence: Figure 7.1 shows that context-based Meta-RL approaches exhibit higher sample efficiency compared with SAC. Context-based Meta-RL converges in approximately 30 episodes (7.5 episodes per building) while SAC requires approximately 60 episodes (15 episodes per building), yet SAC attains stronger or more stable final performance under commercial generalization. Hence, sample efficiency alone does not guarantee superior zero-shot transfer when regime mismatch is substantial.

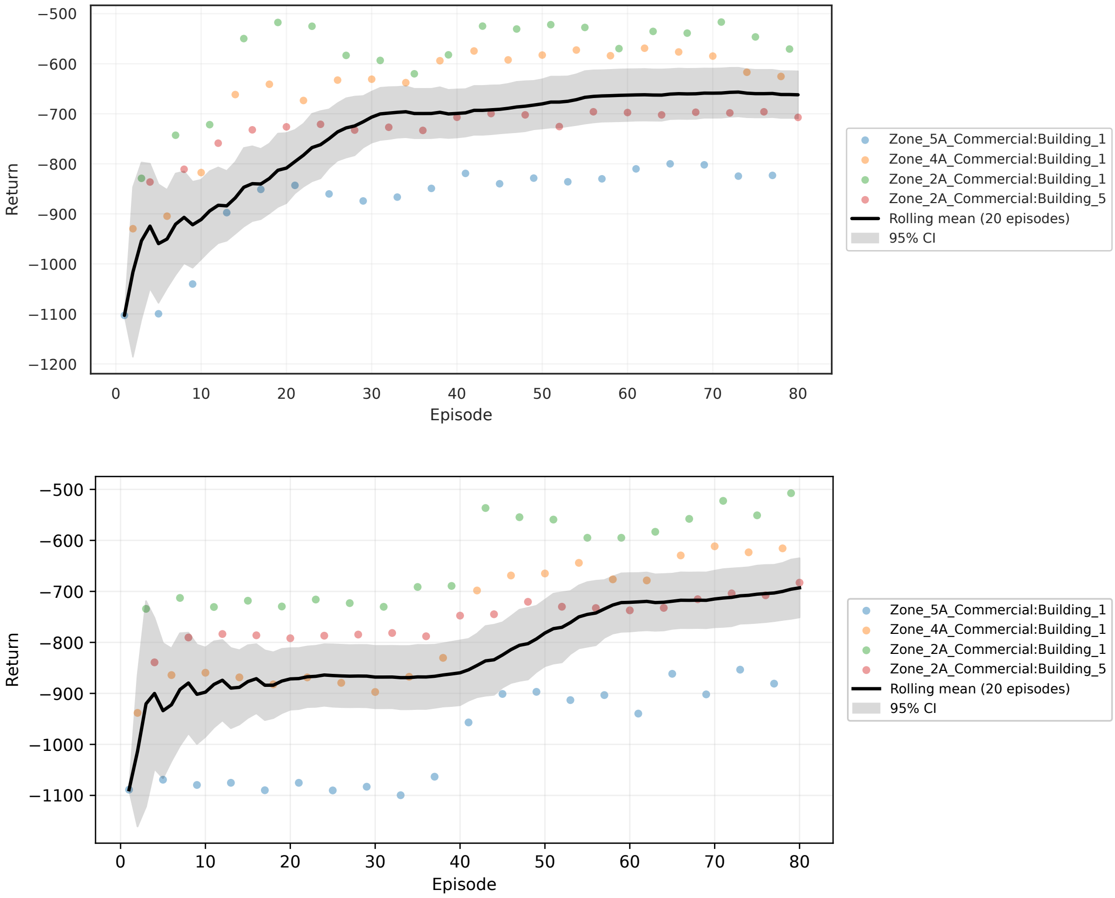


Figure 7.1 Seed 100: Commercial Learning Curves (Return vs. episode index).
Top: Context-based Meta-RL. Bottom: Pretrained SAC.

D. What the setup rewards and what it punishes.

The protocol uses a small number of buildings for training. Consequently, the setup strongly rewards policies that are (i) robust to task mismatch and (ii) immediately effective under distribution shift (zero-shot deployment). Conversely, it penalizes methods whose advantage depends on reliable online task identification and adaptation, since any early miscalibration directly accumulates into a higher electricity bill. This aspect particularly affected Meta-RL methods that weren't given sufficient space for unrewarded exploratory behavior during early environment interactions (few-shot deployment).

E. Highlighted Limitations for Meta-Reinforcement Learning.

The observed degradation of Meta-RL methods on commercial buildings points to several limitations:

(i) meta-overfitting to a narrow subset of training tasks when the number of meta-training buildings is small.

(ii) implicit exploration costs during adaptation, which are undesirable when the objective is an accumulated bill.

(iii) difficulty handling multi-modal task families where optimal policies may differ qualitatively across regimes.

These limitations are consistent with the comparatively high standard deviations and the weaker commercial test means of PEARL and MANN relative to SAC, as well as the failure mode of RL^2.

7.4 Microscopic Analysis: Out-of-Distribution Case Study

Building 4 (Zone 2A) exhibits atypical operational characteristics compared with other buildings in the datasets. In particular, its net power profile resembles that of a prosumer or "power-plant-like" entity rather than a conventional demand-dominant consumer. This results in markedly different reward dynamics relative to buildings drawn from the same training seed, indicating that Building 4 represents an out-of-distribution (OOD) or high-variance case within the training population.

Empirically, SAC achieved substantially higher revenue on Building 4 than all evaluated Meta-RL variants. Because Building 4's returns are extreme relative to the remaining buildings, this single case strongly influenced the overall mean performance metric of the seed η, highlighting the sensitivity of aggregate statistics to anomalous tasks.

A plausible explanation for the inferior PEARL performance on Building 4 is the use of shared parameters and shared optimizers across tasks/buildings as explained in section 5.2. During meta-training, gradients from multiple buildings are aggregated into a single update step. This encourages the global parameters to drift toward solutions that generalize across the majority of buildings. However, such aggregation can attenuate task-specific learning signals and may implicitly discourage exploration strategies that are beneficial for rare or anomalous environments. Consequently, the PEARL agent may under-explore the unique operating regime of Building 4 and miss high-reward opportunities that require stronger deviation from the general population policy.

SAC's generalization across buildings and operating modes remains contingent on the diversity of training data. When SAC encounters an atypical case during training, it may learn behaviors that remain effective under larger distribution shifts. Subsequently, SAC also outperformed PEARL in testing on a similar held-out building (Building 4 - Zone 4A), suggesting that explicit per-task learning without cross-task gradient averaging can be advantageous when task heterogeneity is high.

Split	Building	Algorithm	
		Pretrained SAC	PEARL
Eval	Zone_2A_Commercial:Building_4	-2.671	-1.912
	Zone_2A_Commercial:Building_5	0.899	0.921
	Zone_3A_Commercial:Building_5	0.893	0.912
	Zone_5A_Commercial:Building_5	0.896	0.930
	η	0.004	0.213
Test	Zone_4A_Commercial:Building_1	0.739	0.812
	Zone_4A_Commercial:Building_4	-0.501	-0.095
	Zone_5A_Commercial:Building_1	0.780	0.853
	Zone_5A_Commercial:Building_4	0.250	0.521
	η	0.317	0.522

Table 7.4 Commercial Experiment (Seed 17). The Mean values of the Net Electricity Bill KPI. μ SAC vs μ PEARL.

7.5 Zooming in on KPIs Diagnostics for a Representative Seed

Figure 7.2 summarizes the performance of the trained Context-based Meta-RL agent relative to the RBC baseline across buildings and KPIs for the random seed "42". The y-axis lists the individual buildings. The upper block (first four rows) corresponds to the buildings sampled during training; the reported values for these buildings are obtained by running a single "Evaluation" episode. The bold horizontal separator delineates the transition to held-out (unseen) buildings, for which the values are computed from a single "Testing" episode executed using the same trained policy. The x-axis enumerates the KPIs defined in chapter 6 (e.g., imports/exports, electricity cost components, peak-related metrics, ramping, load-factor proxy, and carbon emissions).

Each cell reports the percentage change of the RL agent with respect to RBC for the *same building* and *episode type* (evaluation or testing), defined as

$$\Delta(\%) = \left(\frac{X_{\text{agent}} - X_{\text{RBC}}}{X_{\text{RBC}}} \right) \times 100, \tag{66}$$

where X_{agent} denotes the value of the corresponding KPI achieved by the RL agent and X_{RBC} denotes the value achieved by RBC under identical conditions.

The color scale encodes both the sign and the magnitude of $\Delta(\%)$. Color intensity is used as a visual aid to interpret the direction and magnitude of the relative change with respect to RBC: cells can become increasingly red as the KPI increases, or vice versa, depending on the interpretation of the change.

For example, an increase in Revenue from exports is green, indicating higher feed-in income (economically beneficial), while on the other side, an increase in Exports is red, corresponding to more energy injected into the grid and implying lower on-site self-consumption.

Accordingly, positive deltas in Imports indicate increased reliance on the grid energy, whereas negative deltas indicate reduced grid dependency. Likewise, positive deltas in Daily Peak Average, Ramping, and $1 -$ Load Factor reflect higher demand peaks, faster power fluctuations, and less uniform demand profiles, respectively (i.e., reduced grid-friendliness) while negative deltas indicate improvement in these dimensions.

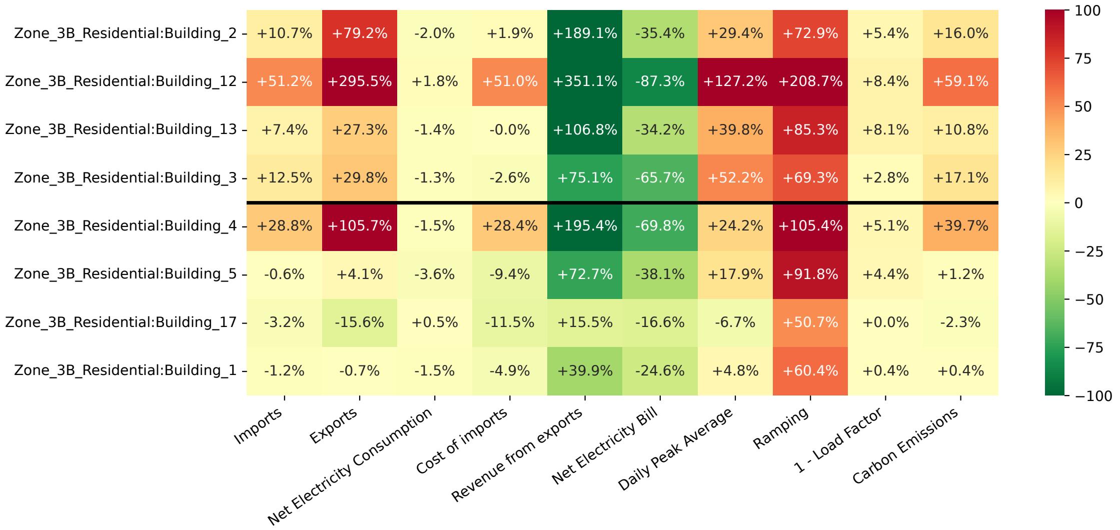


Figure 7.2 KPIs $\Delta(\%)$ Heatmap for "Evaluation / Seen" and "Testing / Unseen" buildings

The results indicate that the learned agent achieves a clear improvement over the RBC baseline with respect to the primary objective. In particular, the Net Electricity Bill is reduced substantially relative to RBC across both evaluation and held-out buildings, demonstrating that the policy generalizes cost-saving behavior beyond the training set.

At first glance, the joint pattern of increased grid Imports and Exports could be interpreted as weaker utilization of local generation. However, the observed behavior is better explained by a more efficient use of the storage system. The agent performs price-driven load shifting: it increases imports during low-price hours to charge the battery and later discharges during high-price periods, thereby reducing the electricity bill while keeping Net Electricity Consumption approximately unchanged compared to RBC. In other words, the improvement is primarily achieved through temporal arbitrage rather than changes in total energy demand.

This strategy comes with trade-offs from a grid-operation perspective. The results show notable increases in peak-related metrics and Ramping, which suggests a less grid-friendly profile with sharper changes in power exchange. Nevertheless, these effects highlight a potential lever for system-level coordination: grid operators could adapt time-varying tariffs or introduce price signals that internalize network constraints, incentivizing the agent to avoid operating regimes that create excessive peaks while still enabling economically efficient flexibility.

Finally, the increases in Carbon Emissions are consistent with the single-objective formulation adopted in this experiment, which focuses on cost minimization. This motivates future work in multi-objective control, where economic performance is optimized jointly with grid-friendliness and emissions, enabling the discovery of policies that balance these competing targets and operate within practical operational limits.

Chapter 8

Conclusion and Future Work

This chapter presents a summary of the key findings from this study and provides recommendations on how to build on those key findings.

8.1 Conclusion

This thesis explored whether Meta-RL can enable adaptable Building Energy Management Systems that perform well when deployed on *previously unseen* buildings. The study compared three Meta-RL families (PEARL, MANN, RL^2) against a strong deep RL baseline (pretrained SAC) and a RBC, using a KPI suite centered on the net electricity bill and complemented by grid- and sustainability-relevant metrics.

A central finding from chapter 7 is that training one joint model over residential and commercial buildings did not generalize reliably across types. The main driver is a reward-scale mismatch: commercial demand magnitudes (and therefore cost signals) dominate training, which can harm residential performance. Separating the task families produced more stable learning and clearer generalization behavior.

Within the residential setting, learning-based controllers consistently improved the net bill relative to RBC. PEARL and pretrained SAC achieved the strongest average results on held-out buildings, while RL^2 and MANN still delivered gains but were less competitive. Notably, the performance gap between evaluation (seen) and testing (unseen) buildings was comparatively small, suggesting that residential tasks formed a smoother distribution where transferable policies can generalize effectively in a zero-shot manner.

Within the commercial setting, generalization was harder and variance across buildings/seeds was larger. Pretrained SAC achieved the best average performance, while Meta-RL methods degraded more noticeably, and RL^2 exhibited a clear failure mode on some buildings. KPI analysis also showed that cost savings largely came from storage-driven temporal arbitrage, but a cost-only objective can increase peaks/ramping and may worsen emissions depending on grid conditions. The out-of-distribution case study further highlighted that strong task heterogeneity can bias shared meta-training toward majority regimes, reducing robustness on atypical buildings.

Overall, the results suggest that Meta-RL can be effective for adaptable BEMS in smoother task families, but robust zero-shot deployment in heterogeneous commercial scenarios remains challenging without stronger mechanisms for normalization, regime handling, and constraint-awareness.

8.2 Recommendations for Future Research

While this thesis demonstrated the feasibility of applying Meta-RL for adaptable BEMS, the obtained results also indicate that performance can be sensitive to the **meta-training task distribution** and the **evaluation protocol**. Future work should therefore focus on improving Meta-RL generalization by expanding meta-training to cover a broader and more representative set of tasks (e.g., increasing the number of meta-training buildings and including more diverse operational modes) in order to reduce meta-overfitting to a narrow subset of building behaviors. In addition, future studies should investigate the effect of providing a longer context window before control (i.e., allowing more observations for task inference prior to executing actions) or alternatively evaluating Meta-RL under a few-shot adaptation setting rather than strict zero-shot deployment, to better reflect realistic commissioning scenarios.

This thesis was also limited in the set of controllable actions considered, which primarily emphasized battery scheduling. A natural extension is to include additional controllable components such as heat pump operation, thermal energy storage, and domestic hot water (DHW) supply. Incorporating these actuators would increase flexibility and enable the controller to exploit both electrical and thermal domains. More importantly, it would allow investigating whether Meta-RL can truly adapt to whichever controllable devices exist in a given building environment and still achieve reliable performance.

Finally, the optimization objective in this thesis was largely driven by electricity cost, which can unintentionally lead to undesirable grid-side effects such as increased peaks, ramping, or emissions. Future research should move toward multi-objective and constrained control by incorporating additional objectives and constraints. This can be addressed using constrained RL, Lagrangian formulations, or Pareto optimization approaches, with the goal of producing controllers that are not only economically efficient but also grid-friendly and aligned with sustainability targets.

Appendix A

Literature Research

	Climate 1 (%)	Climate 2 (%)	Climate 3 (%)	Climate 4 (%)
No Control	109.34	114.13	100.73	105.19
RBC	**100.00**	**100.00**	**100.00**	**100.00**
Random initialization	101.18 ± 2.31	103.59 ± 1.98	111.51 ± 2.96	104.26 ± 1.29
MAML	103.91 ± 0.86	98.91 ± 0.45	99.12 ± 2.42	97.02 ± 1.50
RL-MPC	98.22 ± 0.59	100.53 ± 0.41	98.67 ± 1.69	101.20 ± 3.51
Pretrained	99.45 ± 1.32	98.72 ± 0.90	101.35 ± 1.87	99.62 ± 2.20
MetaEMS	$\mathbf{94.73 \pm 1.24}$	$\mathbf{92.65 \pm 0.81}$	$\mathbf{94.13 \pm 1.67}$	$\mathbf{96.65 \pm 0.82}$

Table A.1 Summary of average cost. Each result is the average of three buildings. [56]

Metrics	**Ramping**	**1-load factor**	**Avg. daily peak**	**Peak Elec. demand**	**Net Elec. consumption**
MAML	0.83	0.98	0.97	0.96	1.01
Random initialization	1.01	0.95	1.03	0.95	1.04
RL-MPC	1.00	0.99	1.02	0.97	1.02
Pretrained	0.99	0.94	0.98	0.97	0.99
MetaEMS	**0.78**	**0.98**	**0.83**	**0.91**	**0.98**
Improvement	6.01%	\	13.58%	3.80%	2.02%

Table A.2 Break down of cost by individual objectives on CityLearn environment. Each result is the average of three buildings and four climate zones [56].

Appendix B

Hyperparameters

Symbol	Meaning/definition	Value
Shared hyperparameters (used in multiple methods)		
gamma	Discount factor	0.99
tau	Polyak averaging / target smoothing factor	0.01
batch_size	Mini-batch size	256
replay_size	Replay buffer capacity	200000
min_buffer_size	Minimum buffer size before updates	256
target_entropy_scale	Target entropy scale	1.0
log_STD_min	Policy log-std clamp minimum	-5.0
log_STD_max	Policy log-std clamp maximum	2.0
inner_actor_lr	Actor learning rate	1e-3
inner_critic_lr	Critic learning rate	1e-3
inner_alpha_lr	Temperature (alpha) learning rate	1e-3
Context-based Meta-RL		
z_dim	Latent context dimension	8
context_batch_size	Context inference batch size	128
context_update_interval	Steps between context updates	1
kl_coeff	KL regularization coefficient for context	0.01
use_deterministic_z	Use deterministic context latent	False
context_encoder_hidden_sizes	Context encoder hidden layer sizes	[128, 128]
logvar_clip_min	Context encoder log-variance clamp min	-10.0
logvar_clip_max	Context encoder log-variance clamp max	10.0
actor_hidden_sizes	Actor hidden layer sizes	[256, 256]
critic_hidden_sizes	Critic hidden layer sizes	[256, 256]

Table B.1 Shared Hyperparameters used across all Meta-RL algorithms and SAC algorithm. Followed by Context-based specific Hyperparameters.

Symbol	Meaning/definition	Value
Memory-based Meta-RL		
memory_size	Episodic memory capacity (entries)	512
memory_key_dim	Memory key dimension	32
memory_value_dim	Memory value dimension	32
memory_read_topk	Top-k reads from memory	8
memory_write_interval	Steps between memory writes	167
batch_size	Mini-batch size	256
replay_size	Replay buffer capacity	200000
target_entropy_scale	Target entropy scale	1.0
min_buffer_size	Minimum buffer size before updates	256
memory_encoder_hidden_sizes	Memory encoder hidden layer sizes	[128, 128]
actor_hidden_sizes	Actor hidden layer sizes	[256, 256]
critic_hidden_sizes	Critic hidden layer sizes	[256, 256]
Blackbox / Recurrent Meta-RL		
rnn_hidden_dim	GRU hidden size	128
sequence_length	Sequence length for training	16
burn_in	Burn-in steps excluded from loss	0
bptt_truncation	BPTT truncation length	16
Pretrained SAC RL		
buffer_size	Replay buffer size	100000
batch_size	Mini-batch size	256
tau	Target smoothing coefficient	0.005
gamma	Discount factor	0.99
train_freq	Training frequency (steps)	1
gradient_steps	Gradient steps per update	1
ent_coef	Entropy coefficient setting	auto
target_update_interval	Target network update interval (steps)	1
log_std_init	Policy log-std initialization	-2.0
policy	SAC policy architecture	MlpPolicy

Table B.2 Algorithm-specific Hyperparameters.

Appendix C

Subsidiary Simulation Plots

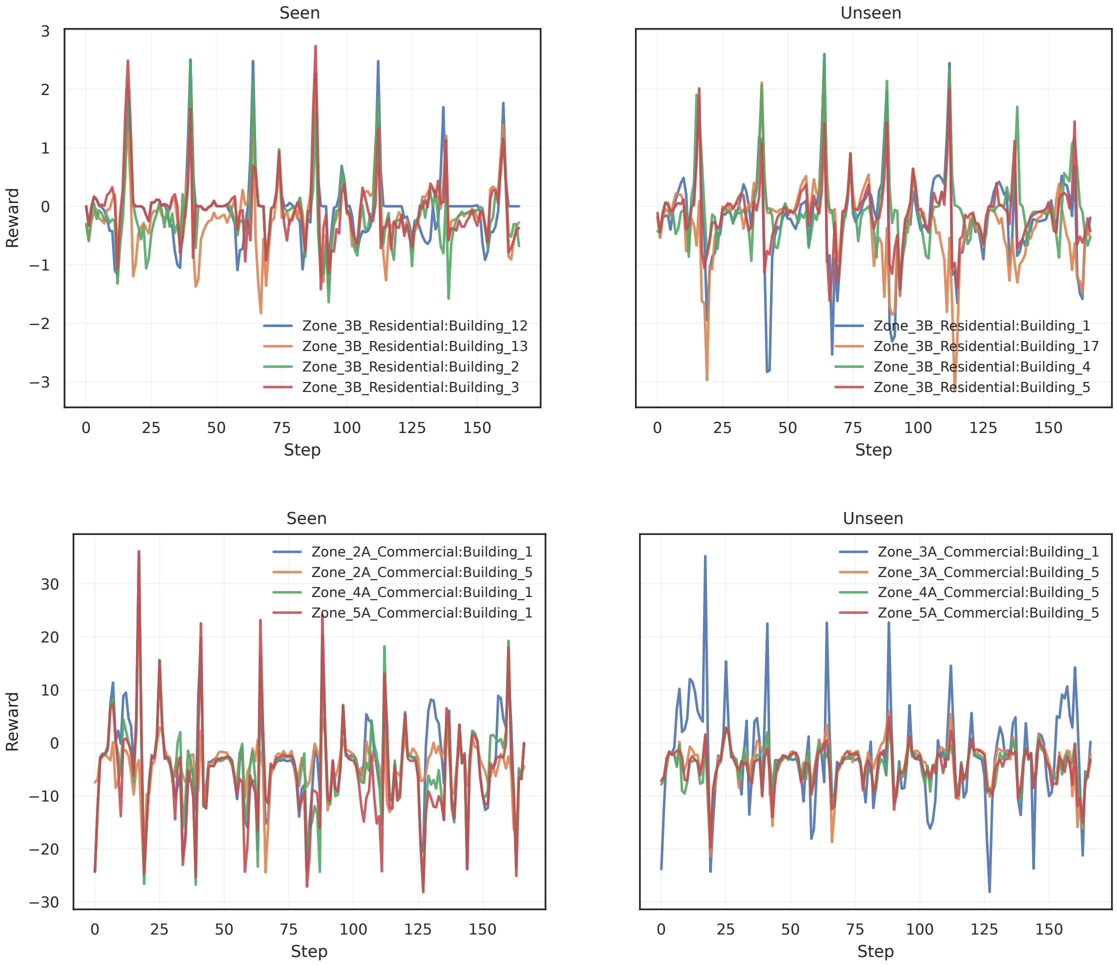
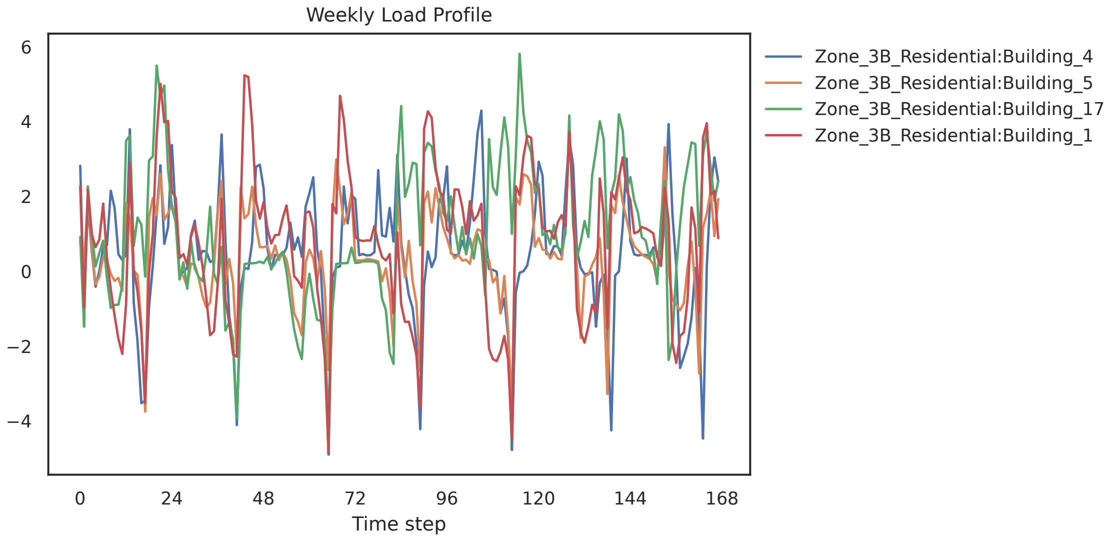


Figure C.1 Context-based Meta-RL performance. Episode Reward vs timestep.
Top: Residential (Seed 42). Bottom: Commercial (Seed 100).



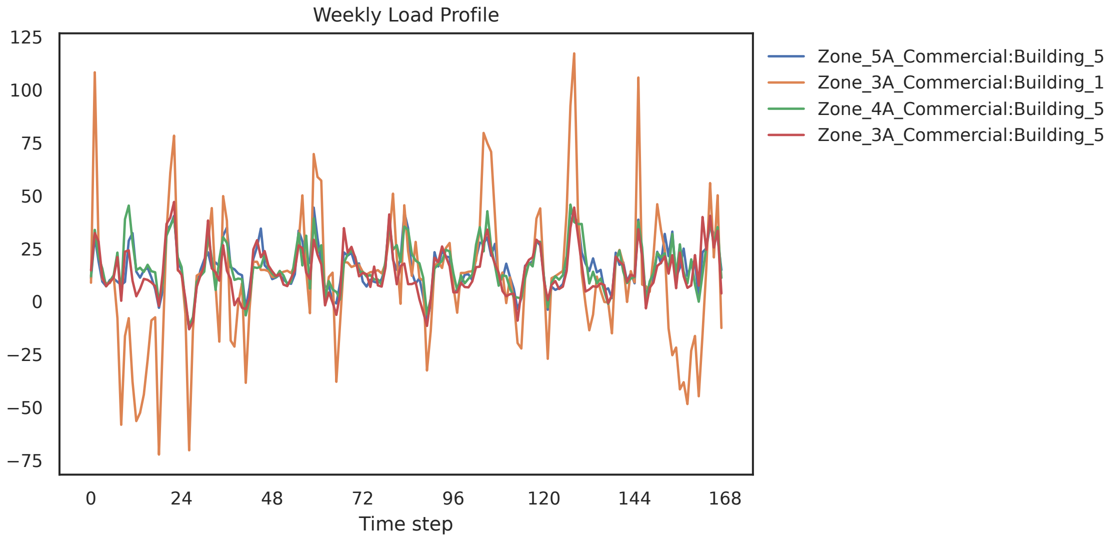


Figure C.2 Performance Results of Context-based Meta-RL.
Weekly Load Profiles for "Test / Unseen" Buildings.
Top: Residential (@ Seed = 42). Bottom: Commercial (@ Seed = 100).

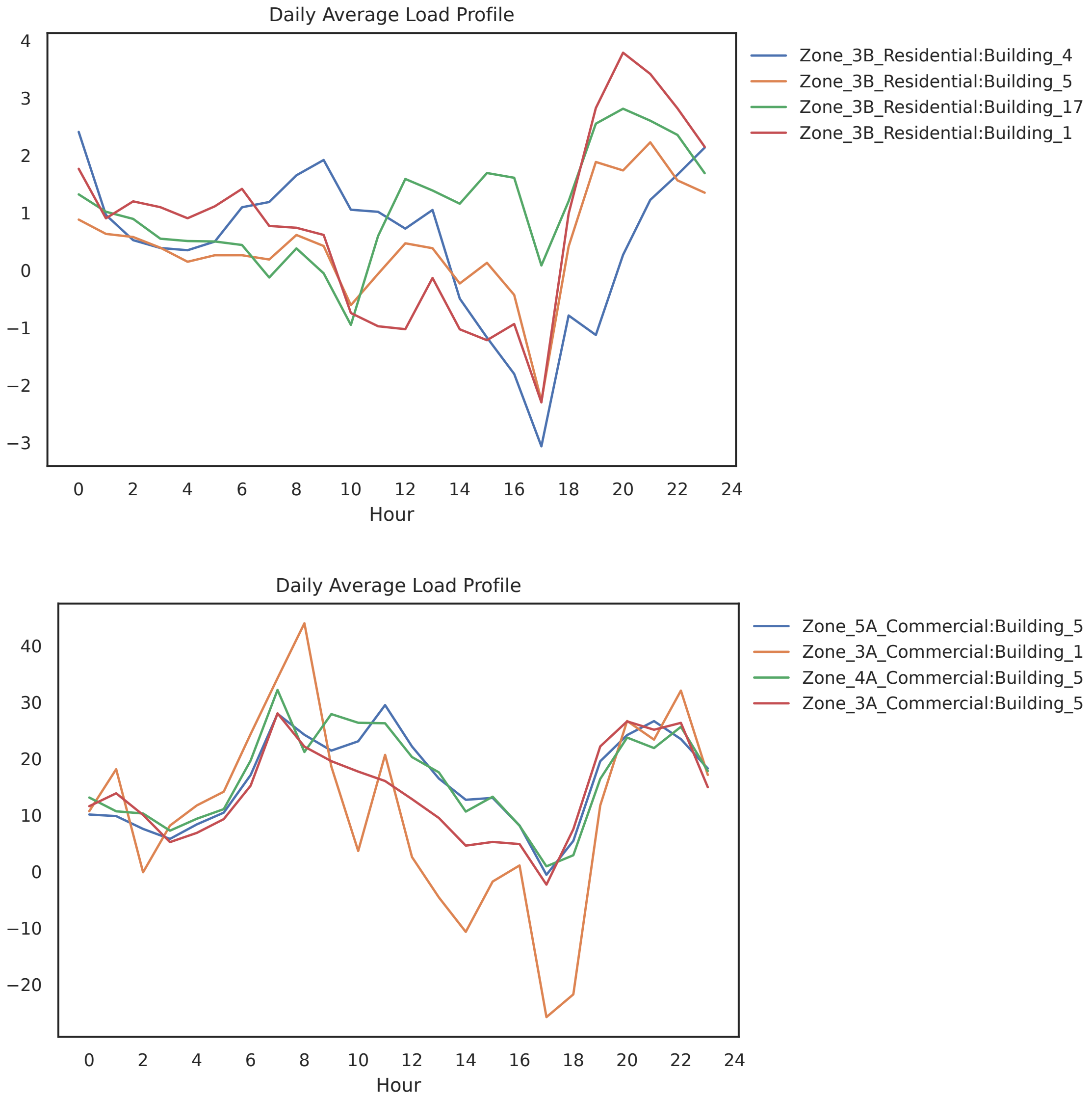


Figure C.3 Performance Results of Context-based Meta-RL.
Average Daily Load Profiles for "Test / Unseen" Buildings.
Top: Residential (@ Seed = 42). Bottom: Commercial (@ Seed = 100).

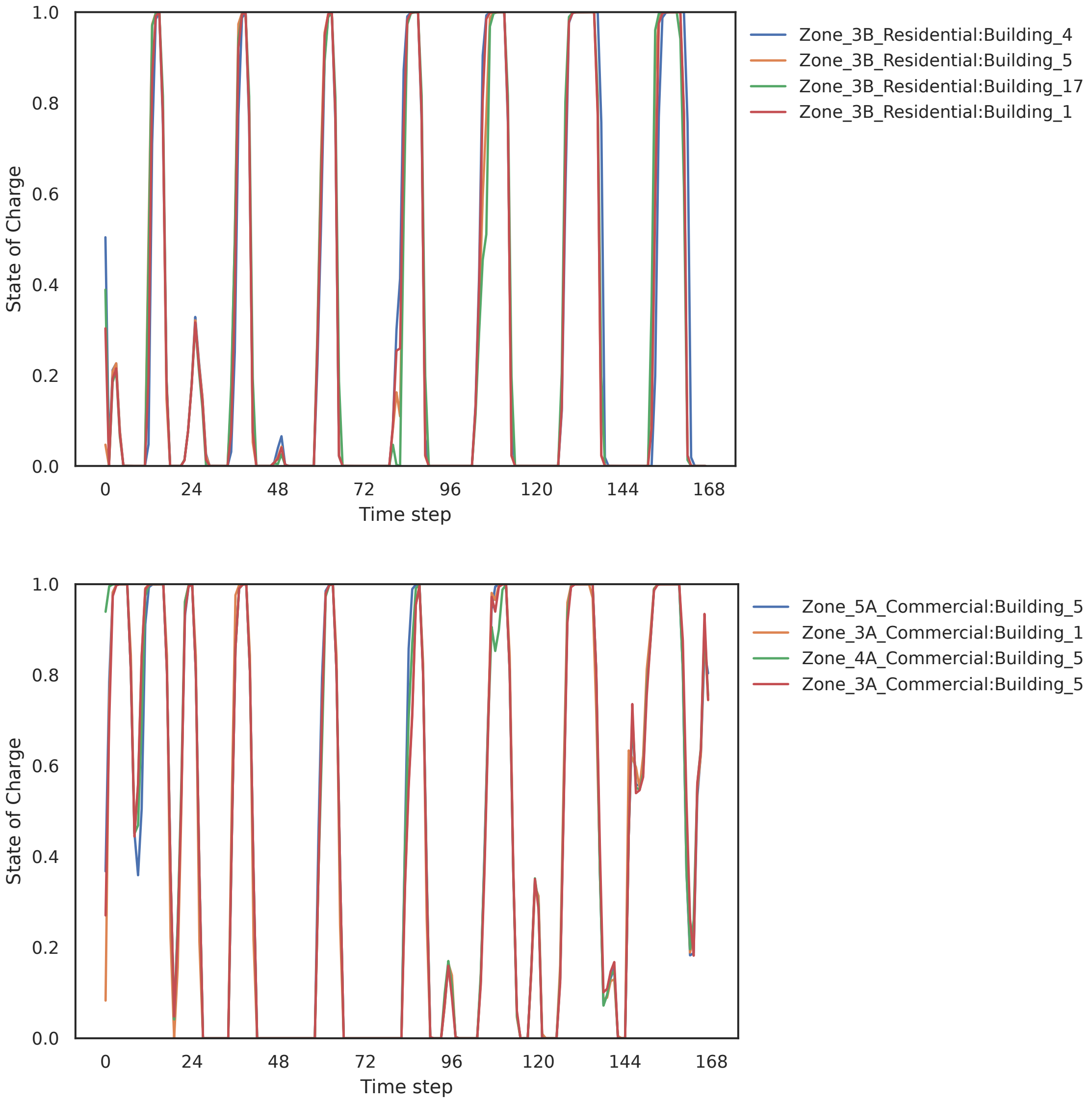


Figure C.4 Performance Results of Context-based Meta-RL.
Battery SOC against timestep for "Test / Unseen" Buildings.
Top: Residential (@ Seed = 42). Bottom: Commercial (@ Seed = 100).

Bibliography

[1] A. Afram and F. Janabi-Sharifi. "Theory and applications of HVAC control systems – A review of model predictive control (MPC)". In: *Building and Environment* 72 (2014), pp. 343–355. DOI: 10.1016/j.buildenv.2013.11.016 (cit. on p. 7).

[2] Agora Energiewende. *Germany's Path to Climate Neutrality by 2045*. 2022. URL: https://www.agora-energiewende.de (visited on 12/28/2025) (cit. on p. 6).

[3] Alexander A. Alemi et al. "Deep Variational Information Bottleneck". In: *International Conference on Learning Representations*. 2017. DOI: 10.48550/arXiv.1612.00410. URL: https://openreview.net/forum?id=HyxQzBceg (cit. on pp. 16–18).

[4] K. Amasyali and N. M. El-Gohary. "A review of data-driven building energy consumption prediction studies". In: *Renewable and Sustainable Energy Reviews* 81 (2018), pp. 1192–1205. DOI: 10.1016/j.rser.2017.04.095 (cit. on p. 6).

[5] Atit Bashyal et al. "Multi-agent deep reinforcement learning based demand response and energy management for heavy industries with discrete manufacturing systems". In: *Applied Energy* 392 (2025), p. 125990. DOI: 10.1016/j.apenergy.2025.125990. URL: https://www.sciencedirect.com/science/article/pii/S0306261925007202 (cit. on p. 30).

[6] Karl Cobbe et al. "Leveraging Procedural Generation to Benchmark Reinforcement Learning". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 2048–2056. URL: https://proceedings.mlr.press/v119/cobbe20a.html (cit. on p. 2).

[7] R. De Coninck and L. Helsen. "Quantification of flexibility in buildings by cost curves – Methodology and application". In: *Applied Energy* 162 (2016), pp. 653–665. DOI: 10.1016/j.apenergy.2015.10.114 (cit. on p. 8).

[8] Deutsche Energie-Agentur (dena). *Digitalisierung der Energiewende im Gebäudesektor*. 2023. URL: https://www.dena.de (visited on 12/27/2025) (cit. on p. 6).

[9] J. Drgona, J. Arroyo, I. Cupeiro Figueroa, et al. "All you need to know about model predictive control for buildings". In: *Annual Reviews in Control* 50 (2020), pp. 190–232. DOI: 10.1016/j.arcontrol.2020.09.001 (cit. on p. 8).

[10] Ján Drgoňa et al. "All you need to know about model predictive control for buildings". In: *Annual Reviews in Control* 50 (2020), pp. 190–232. DOI: 10.1016/j.arcontrol.2020.09.001 (cit. on p. 9).

[11] Yan Duan et al. "RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning". In: *CoRR* abs/1611.02779 (2016). DOI: 10.48550/arXiv.1611.02779. URL: https://arxiv.org/abs/1611.02779 (cit. on pp. 14–16, 19, 45).

[12] I. Fernández et al. "A review on the integration of rule-based and learning-based approaches in building energy management systems". In: *Energy and Buildings* 250 (2021), p. 111298. DOI: 10.1016/j.enbuild.2021.111298 (cit. on p. 7).

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1126–1135. URL: https://proceedings.mlr.press/v70/finn17a.html (cit. on pp. 14, 15, 19).

[14] Ali Forootani, Mohammad Rastegar, and Mohammad Jooshaki. "An Advanced Satisfaction-Based Home Energy Management System Using Deep Reinforcement Learning". In: *IEEE Access* 10 (2022), pp. 47896–47905. DOI: 10.1109/ACCESS.2022.3172327. URL: https://doi.org/10.1109/ACCESS.2022.3172327 (cit. on p. 2).

[15] Fraunhofer ISE. *Photovoltaics Report 2024*. 2024. URL: https://www.ise.fraunhofer.de (visited on 12/28/2025) (cit. on p. 6).

[16] G. Gholamibozanjani and A. Mahdavi. "Deployment challenges of model-based predictive control in buildings". In: *Journal of Building Performance Simulation* 14.3 (2021), pp. 293–308. DOI: 10.1080/19401493.2020.1859989 (cit. on p. 8).

[17] Isaías Gomes et al. "Recent Techniques Used in Home Energy Management Systems: A Review". In: *Energies* 15.8 (2022), p. 2866. DOI: 10.3390/en15082866 (cit. on p. 1).

[18] Erin Grant et al. "Recasting Gradient-Based Meta-Learning as Hierarchical Bayes". In: *International Conference on Learning Representations (ICLR)*. arXiv:1801.08930. 2018. DOI: 10.48550/arXiv.1801.08930. URL: https://openreview.net/forum?id=BJ_UL-k0b (cit. on p. 15).

[19] Abhishek Gupta et al. "Meta-Reinforcement Learning of Structured Exploration Strategies". In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018, pp. 5302–5311. URL: https://papers.nips.cc/paper_files/paper/2018/hash/4de754248c196c85ee4fbdcee89179bd-Abstract.html (cit. on pp. 14, 16).

[20] Tuomas Haarnoja et al. "Soft Actor-Critic Algorithms and Applications". In: *CoRR* abs/1812.05905 (2018). DOI: 10.48550/arXiv.1812.05905. URL: https://arxiv.org/abs/1812.05905 (cit. on pp. 12, 13).

[21] Tuomas Haarnoja et al. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. arXiv:1801.01290. PMLR, Oct. 2018, pp. 1861–1870. URL: https://proceedings.mlr.press/v80/haarnoja18b.html (cit. on pp. 2, 11–13, 16–18, 53).

[22] Timothy M. Hansen et al. “A Partially Observable Markov Decision Process Approach to Residential Home Energy Management”. In: *IEEE Transactions on Smart Grid* 9.2 (Mar. 2018), pp. 1271–1281. DOI: 10.1109/TSG.2016.2582701 (cit. on p. 31).

[23] Peter Henderson et al. “Deep Reinforcement Learning That Matters”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. arXiv version: 1709.06560. 2018, pp. 3207–3214. DOI: 10.1609/aaai.v32i1.11694. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11694 (cit. on pp. 2, 11–13).

[24] H. Huang et al. “A review on rule-based control strategies for smart buildings: Challenges and future directions”. In: *Journal of Building Performance* 13.1 (2022), pp. 17–31 (cit. on p. 7).

[25] *Indicator: German battery storage capacity increases 50% in 2024 – report*. URL: https://www.cleanenergywire.org/news/german-battery-storage-capacity-increases-50-2024-report (visited on 12/22/2025) (cit. on p. 5).

[26] *Indicator: Share of renewables in gross electricity consumption*. URL: https://www.umweltbundesamt.de/en/indicator-share-of-renewables-in-gross-electricity#at-a-glance (visited on 12/22/2025) (cit. on p. 5).

[27] International Energy Agency (IEA). *Germany 2023: Energy Policy Review*. 2023. URL: https://www.iea.org/reports/germany-2023 (visited on 12/28/2025) (cit. on p. 6).

[28] Doseok Jang et al. “Using Meta Reinforcement Learning to Bridge the Gap between Simulation and Experiment in Energy Demand Response”. In: *CoRR* abs/2104.14670 (2021). DOI: 10.48550/arXiv.2104.14670. URL: https://arxiv.org/abs/2104.14670 (cit. on p. 22).

[29] Tomoya Kawakami et al. “An evaluation and implementation of rule-based Home Energy Management System using the Rete algorithm”. In: *The Scientific World Journal* 2014 (2014), p. 591478. DOI: 10.1155/2014/591478 (cit. on p. 1).

[30] M. Killian and M. Kozek. “Ten questions concerning model predictive control for energy efficient buildings”. In: *Building and Environment* 105 (2016), pp. 403–412. DOI: 10.1016/j.buildenv.2016.05.034 (cit. on p. 7).

[31] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *International Conference on Learning Representations*. 2014. DOI: 10.48550/arXiv.1312.6114. URL: https://arxiv.org/abs/1312.6114 (cit. on p. 17).

[32] Felix Langner et al. “Model predictive control of distributed energy resources in residential buildings considering forecast uncertainties”. In: *Energy and Buildings* 303 (2024), p. 113753. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2023.113753 (cit. on p. 1).

[33] Paulo Lissa et al. “Deep reinforcement learning for home energy management system control”. In: *Energy and AI* 3 (2021), p. 100043. ISSN: 2666-5468. DOI: 10.1016/j.egyai.2020.100043 (cit. on p. 1).

[34] Ulrich Ludolfinger et al. “Transformer Model Based Soft Actor-Critic Learning for HEMS”. In: *2023 IEEE International Conference on Power System Technology (PowerCon)*. Jinan, China: IEEE, 2023, pp. 1–6. DOI: 10.1109/PowerCon58120.2023.10331287 (cit. on p. 30).

[35] Y. Ma et al. “Model predictive control for the operation of building cooling systems”. In: *IEEE Transactions on Control Systems Technology* 20.3 (2012), pp. 796–803. DOI: 10.1109/TCST.2011.2134850 (cit. on p. 8).

[36] MarketsandMarkets. *Building Energy Management System (BEMS) Market — Global Forecast to 2030*. 2023. URL: https://www.marketsandmarkets.com/Market-Reports/building-energy-management-systems-bems-market-591.html (visited on 12/27/2025) (cit. on p. 6).

[37] Nikhil Mishra et al. “A Simple Neural Attentive Meta-Learner”. In: *International Conference on Learning Representations (ICLR)*. arXiv:1707.03141. 2018. DOI: 10.48550/arXiv.1707.03141. URL: https://openreview.net/forum?id=B1DmUzWAW (cit. on p. 16).

[38] Zoltan Nagy et al. “Ten questions concerning reinforcement learning for building energy management”. In: *Building and Environment* 241 (2023), p. 110435. ISSN: 0360-1323. DOI: 10.1016/j.buildenv.2023.110435 (cit. on p. 1).

[39] F. Oldewurtel, D. Sturzenegger, and M. Morari. “Importance of occupancy information for building climate control”. In: *Applied Energy* 101 (2013), pp. 521–532. DOI: 10.1016/j.apenergy.2012.06.014 (cit. on p. 7).

[40] *Quarterly greenhouse gas emissions in the EU*. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Quarterly_greenhouse_gas_emissions_in_the_EU (visited on 12/22/2025) (cit. on p. 5).

[41] Kate Rakelly et al. “Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 5331–5340. URL: https://proceedings.mlr.press/v97/rakelly19a.html (cit. on pp. 14–19, 36).

[42] Roland Berger. *Smart Buildings in Europe: A €300 Billion Opportunity*. 2024. URL: https://www.rolandberger.com (visited on 12/28/2025) (cit. on p. 6).

[43] Jonas Rothfuss et al. “ProMP: Proximal Meta-Policy Search”. In: *International Conference on Learning Representations (ICLR)*. arXiv:1810.06784. 2019. DOI: 10.48550/arXiv.1810.06784. URL: https://openreview.net/forum?id=SkxXCi0qFX (cit. on p. 15).

[44] Adam Santoro et al. “Meta-Learning with Memory-Augmented Neural Networks”. In: *Proceedings of the 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. arXiv:1605.06065. PMLR, 2016, pp. 1842–1850. URL: https://proceedings.mlr.press/v48/santoro16.html (cit. on pp. 16, 40).

[45] Gerrit Schoettler et al. “Meta-Reinforcement Learning for Robotic Industrial Insertion Tasks”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2020, pp. 9728–9735. DOI: 10.1109/IROS45743.2020.9340848. URL: https://doi.org/10.1109/IROS45743.2020.9340848 (cit. on pp. 20, 21).

[46] John Schulman et al. “Trust Region Policy Optimization”. In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. arXiv:1502.05477. Lille, France: PMLR, July 2015, pp. 1889–1897. DOI: 10.48550/arXiv.1502.05477. URL: https://proceedings.mlr.press/v37/schulman15.html (cit. on p. 15).

[47] John Schulman et al. “Proximal Policy Optimization Algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017). DOI: 10.48550/arXiv.1707.06347. URL: https://arxiv.org/abs/1707.06347 (cit. on pp. 2, 10–13, 15).

[48] D. Sturzenegger, D. Gyalistras, V. Semeraro, et al. “Model predictive control of a Swiss office building”. In: *Energy and Buildings* 125 (2016), pp. 253–263. DOI: 10.1016/j.enbuild.2016.04.006 (cit. on p. 8).

[49] Richard S. Sutton and Andrew G. Barto. “The agent–environment interaction in a Markov decision process”. In: *Reinforcement Learning: An Introduction*. 2nd ed. Figure 3.1. Cambridge, MA: The MIT Press, 2018, p. 48 (cit. on pp. 8, 9, 14).

[50] Ananya Unnikrishnan. “Financial News-Driven LLM Reinforcement Learning for Portfolio Management”. In: *arXiv preprint arXiv:2411.11059* (2024). DOI: 10.48550/arXiv.2411.11059. URL: https://arxiv.org/abs/2411.11059 (cit. on p. 10).

[51] José R. Vázquez-Canteli and Zoltán Nagy. “Reinforcement learning for demand response: A review of algorithms and modeling techniques”. In: *Applied Energy* 235 (2019), pp. 1072–1089. DOI: 10.1016/j.apenergy.2018.11.002 (cit. on p. 2).

[52] José R. Vázquez-Canteli and Zoltán Nagy. “Reinforcement learning for demand response: A review of algorithms and modeling techniques”. In: *Applied Energy* 235 (2019), pp. 1072–1089. DOI: 10.1016/j.apenergy.2018.11.002 (cit. on p. 9).

[53] José R. Vázquez-Canteli et al. “CityLearn v1.0: An OpenAI Gym Environment for Demand Response with Deep Reinforcement Learning”. In: *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys ’19)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 356–357. DOI: 10.1145/3360322.3360998. URL: https://doi.org/10.1145/3360322.3360998 (cit. on pp. 26, 50, 51).

[54] Jane X. Wang et al. “Learning to reinforcement learn”. In: *arXiv preprint arXiv:1611.05763* (2016). DOI: 10.48550/arXiv.1611.05763. URL: https://arxiv.org/abs/1611.05763 (cit. on pp. 14–16, 19).

[55] Chiyuan Zhang et al. “A Study on Overfitting in Deep Reinforcement Learning”. In: *CoRR* abs/1804.06893 (2018). DOI: 10.48550/arXiv.1804.06893. URL: https://arxiv.org/abs/1804.06893 (cit. on pp. 2, 11–13).

[56] Huiliang Zhang, Di Wu, and Benoit Boulet. “MetaEMS: A Meta Reinforcement Learning-based Control Framework for Building Energy Management System”. In: *CoRR* abs/2210.12590 (2022). DOI: 10.48550/arXiv.2210.12590. URL: https://arxiv.org/abs/2210.12590 (cit. on pp. 21, 22, 31, 69).

[57] Yang Zhang, Huiwen Yan, and Mushuang Liu. “Directed-MAML: Meta Reinforcement Learning Algorithm with Task-directed Approximation”. In: *CoRR* abs/2510.00212 (2025). DOI: 10.48550/arXiv.2510.00212. URL: https://arxiv.org/abs/2510.00212 (cit. on p. 35).

[58] Zhiang Zhang and Khee Poh Lam. “Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system”. In: *Proceedings of the 5th Conference on Systems for Built Environments (BuildSys 2018), Shenzhen, China, November 07–08, 2018*. ACM, 2018, pp. 148–157. DOI: 10.1145/3276774.3276775. URL: https://doi.org/10.1145/3276774.3276775 (cit. on p. 2).