Large Language Model Guided Reinforcement Learning for Heat Pump Control in Smart Buildings

Gökhan Demirel[1], Natascha Fernengel[1], Hallah Butt[1], Natalie Rapp[1], Kevin Förderer[1], Veit Hagenmeyer[1]
[1]Institute for Automation and Applied Informatics (IAI), Karlsruhe Institute of Technology, Germany
{goekhan.demirel, natascha.fernengel, hallah.butt, natalie.rapp, kevin.foerderer, veit.hagenmeyer}@kit.edu

***Abstract*—Heat pumps are essential for the decarbonization of the heating sector but often operate inefficiently due to volatile electricity prices, forecast uncertainties, and diverse comfort preferences. Traditional rule- and model-based controllers ensure stable operation but lack adaptability under dynamic conditions, while data-driven reinforcement learning (RL) provides decision-making flexibility at the cost of interpretability and transparency. This paper presents a hybrid high-level control framework in which a large language model (LLM) serves as a meta-policy, supervising a portfolio of RL-based low-level controllers for residential heat pump operation, formulated as a partially observable Markov decision process (POMDP). The LLM processes structured JSON-formatted prompts containing building states, short-horizon forecasts, candidate actions, and one-step rewards, and outputs either a discrete controller selection (guided mode) or convex weights for action fusion (weighted mode). Benchmark experiments in a realistic building simulation testbed compare four state-of-the-art LLMs (Mistral-7B, Llama-3.1-8B, DeepSeek-Chat, and GPT-4.1) against standalone RL (PPO, SAC, DDPG, and A2C). A two-dimensional decoding sensitivity analysis identifies balanced stochastic decoding hyperparameter regions (temperature and nucleus sampling) that prevent response errors, thereby stabilizing control performance. Results show that LLM-guided RL achieves up to 5.1% higher mean reward over the best standalone RL controller.**

***Index Terms*—Decision-making, heat pumps, hybrid control, large language models (LLMs), reinforcement learning.**

I. Introduction

Decarbonizing residential heating increasingly relies on the adoption of heat pumps (HP). Large-scale field studies have shown that incorrectly configured heating curves and setpoints lead to degraded efficiency, indicating the need for high-level control [1], [2]. Conventional low-level controllers provide fast, stable responses and can be tuned for seasonal comfort or cost preferences. However, a high-level planner is required to balance multi-objective trade-offs and adapt strategy selection to seasonal and contextual variations.

Dynamic day-ahead pricing and demand-response schedules increase volatility. HP control must maintain comfort, exploit thermal storage, adapt to forecasts, and remain interpretable to stakeholders. This yields a partially observable, stochastic, long-horizon control problem [3]. This work employs a hybrid architecture in which a high-level planner selects or combines low-level controllers [2]. In this context, large language models (LLMs) offers a semantic, context-aware interface for high-level planning. Recent advances in robotics and decision-making show that commercial LLMs can perform high-level planning from structured prompts, demonstrating strong zero-shot generalization (i.e., the ability to solve unseen tasks without additional training) [4]. In energy systems, low-level controllers such as rule-based logic, model predictive control (MPC), and reinforcement learning (RL) ensure safety but often lack the flexibility to align with user goals and coordinate over long horizons. End-to-end RL can learn complex behaviors but faces sparse rewards, long-horizon planning challenges, and high-dimensional action spaces [5], [6]. LLMs can decompose tasks using broad knowledge [4] but are unsuitable for safety-critical low-level control due to possible so-called hallucinations.

This work formulates prompt engineering as a control problem over the stochastic dynamics of LLMs and embeds a LLM planner within a hybrid HP controller that integrates multiple low-level controllers. It outputs either the index of a selected controller or a weighting over the controller set together with a natural-language explanation. This design merges the structure and safety of technical controllers with the adaptability of LLMs. This work addresses three main research questions:

1) Can LLM-driven meta-policies outperform standalone RL controllers in terms of mean cumulative reward?
2) How do decoding parameters influence mean reward, variance, and the rate of invalid responses?
3) Can weighted (action-fusion) meta-policies enhance robustness compared to guided (selection) strategies?

The key contributions of this work are:

- We formalize LLM-guided RL for HP control as a hybrid decision process that integrates structured prompts for high-level planning.
- We benchmark four LLMs (two open-source and two commercial) in realistic building simulations and conduct a decoding-policy sensitivity analysis (temperature, top-p) for guided and weighted meta-policies.

The remainder of this paper is structured as follows: Section II reviews related work on applications of LLMs in energy systems. Section III defines the hybrid HP control problem formulation, while Section IV details the hybrid control architecture and prompt design. Section V presents the experimental setup and performance analysis. Finally, Section VI concludes the paper and outlines future research directions.

II. Related Work

LLMs achieve broad generalization, reasoning, and context understanding through scaling, pre-training, and in-context learning [7], [8]. While their main applications have so far been in domains such as robotics and software engineering, they are increasingly explored in the energy sector for planning, interpretability, and human interaction [9]–[12].

In building-level control, LLMs have been used to improve transparency by generating natural-language justifications for control actions [10], and recent surveys discuss opportunities and challenges [9]. LLM-based interfaces have also been proposed to translate user inputs into structured control parameters for Home Energy Management Systems (HEMS) [12]. At the grid-level, hybrid frameworks that combine LLMs with stochastic unit commitment have shown improved cost-efficiency and temporal adaptability under renewable uncertainty [11], [13]. Domain-specific LLMs fine-tuned for advanced power dispatch achieve near state-of-the-art performance and improve operator support [14]. In power grid simulations, structured prompting improves robustness across diverse environments [15], and local LLM agents can iteratively mitigate voltage violations without fine-tuning [16]. Security concerns such as semantic drift, data leakage, and denial-of-service attacks are highlighted in [17], motivating the restriction of LLMs to high-level planning roles and the use of robust access controls.

To the best of our knowledge, the present work is the first embedding a LLM into a hybrid control pipeline for residential HP, dynamically selecting or combining low-level controllers based on structured prompts. Beyond performance in partially observable environments, we evaluate the informativeness of LLM-generated explanations for transparency and verification [9]. Figure 1 illustrates the proposed architecture. We benchmark four LLMs through a unified OpenAI-compatible API, executed locally or over external providers [18]–[21]. While LLMs have been integrated into energy management systems before, no prior work investigates stochastic policy analysis for control.

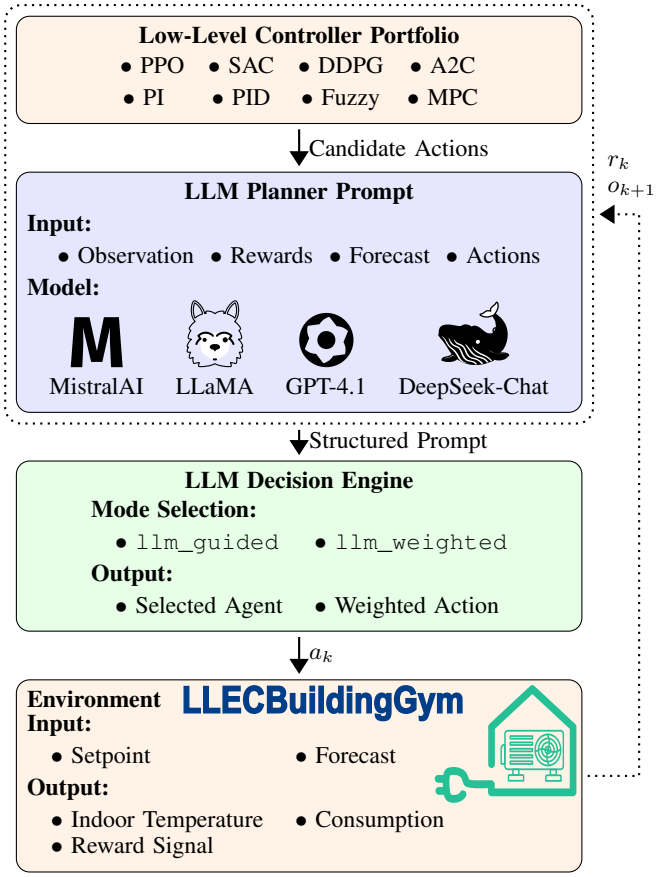

Fig. 1: Hybrid architecture with a LLM planner and low-level controller portfolio Π for heat pump operation.

III. Problem Formulation

A. Low-Level Control Policy Optimization

The *LLECBuildingGym* [3] models a residential building equipped with an air-to-water heat pump. The control problem is formalized as a Partially Observable Markov Decision Process (POMDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{T}_0, R, \gamma)$, where $\mathcal{S}$ and $\mathcal{A} \subseteq [-1, 1]$ represent continuous state and action spaces, respectively. Positive action values correspond to heating, while negative values indicate cooling. The system dynamics are modeled by the transition function $\mathcal{T}(s_{k+1} \mid s_k, a_k)$, representing the probability of transitioning from state $s_k \in \mathcal{S}$ to $s_{k+1} \in \mathcal{S}$ under action $a_k \in \mathcal{A}$. The initial state distribution is denoted by $\mathcal{T}_0(s_0)$, and $\gamma \in [0, 1)$ is the discount factor. The agent interacts with the environment every $\Delta t = 5\,\text{min}$. At each timestep k, it observes a noisy measurement $o_k = s_k + \varepsilon_k$, where $\varepsilon_k \sim \mathcal{N}(0, 0.01)$, selects an action $a_k \in \mathcal{A}$, and receives a scalar reward R_k. The reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is state-dependent and satisfies the bounded reward assumption $R_k \in [-R_{\max}, R_{\max}]$.

$$\mathcal{T}(\xi \mid \pi) = \mathcal{T}_0(s_0) \prod_{k=0}^{H-1} \mathcal{T}(s_{k+1} \mid s_k, a_k)\, \pi(a_k \mid o_k), \quad (1)$$

$$\mathcal{R}(\xi) = \sum_{k=0}^{H-1} \gamma^k R_k, \quad (2)$$

$$\mathbb{E}_{\xi\sim\pi}\left[\mathcal{R}(\xi)\right] = \int_{\xi} \mathcal{T}(\xi \mid \pi)\mathcal{R}(\xi), \quad (3)$$

$$R_{\text{temp}} = \exp\left(-|T_{\text{in},k} - T_{\text{set},k}|\right), \quad (4)$$

$$R_{\text{econ}} = -\frac{p_k \cdot |a_k|}{p_{\max}}, \quad (5)$$

$$R_k = w_{\text{temp}} R_{\text{temp}} + w_{\text{econ}} R_{\text{econ}}, \quad (6)$$

$$\pi^*(a \mid \boldsymbol{o}) = \arg\max_{\pi} \mathop{\mathbb{E}}_{\xi\sim\pi}\Big[\mathcal{R}(\xi)\Big]. \quad (7)$$

A full trajectory $\xi = (s_0, a_0, s_1, \ldots, s_H)$ follows the policy-dependent distribution defined in Eq. (1), which models the joint likelihood over state transitions and policy actions. The trajectory return is computed as the discounted sum of rewards, as shown in Eq. (2). The expected return under policy π is

then expressed as a path integral weighted by the trajectory distribution, as given in Eq. (3). The total scalar reward R_k defined in Eq. (6) combines a temperature comfort term R_{temp} from Eq. (4) and an economic cost term R_{econ} from Eq. (5). In Eq. (4), $T_{\text{in},k}$ and $T_{\text{set},k}$ denote the indoor and target setpoint temperatures, respectively. Eq. (5), p_k represents the electricity price at timestep k under a dynamic tariff. These two components are combined in Eq. (6) through tunable weighting factors w_{temp} and w_{econ} to form the total reward R_k. The planning horizon is fixed at $H = 288$ time steps of 5 minutes, corresponding to one full day. The low-level controllers aim to find the optimal policy π^* that maximizes the expected return in Eq. (7).

B. High-Level LLM-Guided Meta-Policy Planning

To improve robustness and adaptability, we introduce a high-level planner based on a LLM. The planner selects from a portfolio $\Pi = \{\pi^{\text{PI}}, \pi^{\text{PID}}, \pi^{\text{Fuzzy}}, \pi^{\text{MPC}}, \pi^{\text{RL}}\}$ of low-level controllers. This work focuses on π^{RL}, which includes parameterized policies such as PPO [22], SAC [23], DDPG [24], and A2C [25]. These policies are used to analyze the effects of LLM-guided planning. The high-level decision-making control can be viewed as a meta-policy π^{LLM} that maps observation trajectories to controller selections or weight vectors. The overall objective remains to maximize the expected return of the composed policy:

$$\pi^{\text{LLM}*} = \arg\max_{\pi^{\text{LLM}}} \mathbb{E}_{\xi \sim \pi^{\text{LLM}} \circ \Pi} \left[\mathcal{R}(\xi) \right], \tag{8}$$

where $\pi^{\text{LLM}} \circ \Pi$ denotes the hybrid policy induced by LLM-guided selection or combining of the controllers in Π.

C. Decoding Policy as a Stochastic Output Layer

Let V denote the vocabulary and $z_t \in \mathbb{R}^{|V|}$ the logits at step t for context c_t. We define the base distribution as

$$\rho_{\text{base}}(x \mid c_t) \;=\; \frac{\exp\big(z_t(x)\big)}{\sum_{u \in V} \exp\big(z_t(u)\big)} \quad \text{for } x \in V. \tag{9}$$

Temperature sampling rescales the logits before the softmax with the decoding temperature $\tau > 0$:

$$\rho_\tau(x \mid c_t) \;=\; \frac{\exp\big(z_t(x)/\tau\big)}{\sum_{u \in V} \exp\big(z_t(u)/\tau\big)}, \tag{10}$$

where the parameter τ denotes the decoding temperature, a scaling factor that controls sampling randomness; larger values flatten the distribution and increase stochasticity, whereas smaller values sharpen it and yield more deterministic outputs [26].

For nucleus (top-p) sampling we first sort all tokens $x \in V$ in descending order of $\rho_{\text{base}}(x \mid c_t)$. Starting from the most probable token we iteratively add tokens to a cumulative set $S_p(c_t)$ until the cumulative probability mass $\sum_{u \in S_p(c_t)} \rho_{\text{base}}(u \mid c_t) \geq p$ is reached. This procedure yields a fixed finite subset $S_p(c_t) \subseteq V$ which is then used for renormalization as

$$\rho_p^{\text{nuc}}(x \mid c_t) = \begin{cases} \dfrac{\rho_{\text{base}}(x \mid c_t)}{\sum_{u \in S_p(c_t)} \rho_{\text{base}}(u \mid c_t)}, & x \in S_p(c_t), \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where nucleus sampling discards low-probability tokens and renormalizes the remaining probabilities such that they sum to one, effectively redistributing the probability mass among the most likely candidates [27]. For instance, if the vocabulary consists of 100 tokens, only the most probable subset whose cumulative probability exceeds p (e.g., 90%) is retained, and the corresponding probabilities are rescaled to sum to one. This definition is nonrecursive since $S_p(c_t)$ is constructed from ρ_{base} and remains fixed during renormalization [27]. The resulting stochastic decoding distribution induces variability in the LLM meta-policy π^{LLM} that maps decision prompts to controller selection or weighting (see Sec. V).

IV. Implementation

This section outlines the LLM-guided hybrid control framework, including control modes, prompt design, and the hybrid control loop.

A. LLM Control Modes

The LLM operates as a high-level planner above a portfolio of $|\Pi|$ pre-trained low-level controllers. At each timestep k, each controller $\pi^i \in \Pi$ proposes a candidate action $a_k^{(i)}$. The LLM receives a structured prompt encoding observations and returns either the index of a selected controller $\pi^{(k)} \in \Pi$ or a convex combination of actions. Using the prompt structure described in Section IV-B, the LLM operates in one of two modes (see Table I).

1) Guided Mode: The LLM selects the best-scoring controller index $i_k^\star = \arg\max_i \; \text{score}(\pi^i, s_k)$ with $\pi^i \in \Pi$, and applies the action

$$a_k = a_k^{(i_k^\star)}. \tag{12}$$

2) Weighted Mode: The LLM selects two controllers $\pi_{i_1}, \pi_{i_2} \in \Pi$ based on its internal decision process. It assigns nonnegative weights $w_{k,i_1}, w_{k,i_2} \geq 0$ with $w_{k,i_1} + w_{k,i_2} = 1$. The action is then computed as

$$a_k = w_{k,i_1}\, a_k^{(i_1)} + w_{k,i_2}\, a_k^{(i_2)}. \tag{13}$$

TABLE I: Overview of LLM control modes.

#	Mode	Output		Semantics
1	Guided	Eq. (12)	`{"chosen_algo": i}`	select π^i from Π
2	Weighted	Eq. (13)	`{"weights": {i: w}}`	weighted actions $w_i \geq 0$, $\sum_i w_i = 1$

Listing 1. System and task prompts for guided and weighted modes in the hybrid LLM planner for HP control.

```
ROLE: Heat pump supervisor making control choices.
INPUT JSON keys:
 s:int (step >= 0); indt:int (indoor degC * 10); st:int (
     setpoint degC * 10);
 p_forecast:list[int] (next-H prices EUR/kWh * 1000);
     tset_forecast:list[int] (next-H setpoints degC * 10);
 can:dict[algo->{power,reward,reward_temperature,
     reward_economic,action}];
 pr:int (prev reward * 100); cr:int (cum reward * 100).
HARD OUTPUT RULES:
 - Return JSON ONLY (minified). No prose, no markdown fences
     , no backticks.
 - Use double quotes only. No NaN/Infinity. No extra fields.
 - Do NOT prefer an algorithm because of examples; decide
     only from provided metrics.
 - Keep `reason` short, single line, ASCII only.
 - Optional <JSON>...</JSON> tags may be used and are
     ignored by the parser.
TASK (llm_guided):
 Return {"chosen_algo": one_of(["ppo","sac","ddpg","a2c"]),
     "reason":"short"}.
TASK (llm_weighted):
 Return {"weights":{ALGO:float,...},"reason":"short"}.
 - Allowed ALGO keys: ["ppo","sac","ddpg","a2c"].
 - Exactly two weights must be non-zero.
 - Weights must be non-negative and sum to 1.0.
```

B. LLM Prompt Design

We adopt a two-message prompting scheme consisting of a fixed initial system prompt and step-wise decision prompts. The initial system prompt ($\sim$1k characters) defines the task and requires a JSON object with: step index, indoor temperature, temperature setpoint, electricity price forecast, setpoint forecast, candidate actions, previous, and cumulative reward. The second message type is transmitted at each timestep k and contains a JSON structure with the current state, forecasts, candidate actions, and rewards. All numerical values are compactly encoded (temperatures $\times 10$, prices $\times 1000$, rewards $\times 100$), reducing token usage by $\approx 30\%$ compared with unscaled floating-point strings. The LLM interaction is stateless, with each decision based only on the latest prompt without preserving prior interaction history.

The final prompt structure was derived through iterative refinement, prioritizing JSON validity, decision stability, and minimal token usage across all evaluated models. Table I summarizes the output structure for the guided and weighted modes. The structured format in Listing 1 follows the common design conventions of LLM interfaces, separating system and user messages with explicit role declarations consistently used across all experiments. Instead of a free-form textual description, the system prompt defines a machine-readable instruction block that enforces strict syntax and output constraints, ensuring deterministic JSON responses required for real-time integration. The prefix `ROLE:` serves as a semantic equivalent to the system message, fixing the model's operational context `Heat pump supervisor` and reducing output variance by improving compliance with the specified output rules.

Alternative natural-language prompts resulted in higher output variance and more frequent JSON violations [28]. Consistent with StructuredRAG [29], we found empirically in our experiments that role-based prompts with explicit schema and output rules improved structured-output reliability. Unless stated otherwise, all experiments use a decoding temperature of $\tau = 0.8$ and nucleus sampling with $p = 0.94$, as identified in Section V-B.

C. Hybrid Control Loop

The hybrid control process for a horizon of length H begins with the LLM receiving the structured system and task prompts followed by the sequential execution of the following control steps:

1) **State and Forecast Acquisition:** At each timestep $k \in \{0, \dots, H-1\}$, the environment provides the current observation o_k and a forecast vector $\hat{\mathbf{o}}_{k:k+H-1}$.
2) **Candidate Action and Reward Estimation:** Each controller $\pi^i \in \Pi$ generates a candidate action $a_k^{(i)}$ and the corresponding one-step reward $R_k^{(i)}$.
3) **Prompt Construction:** The observation, forecast, candidate actions, and rewards are encoded into a structured JSON decision prompt u_k following the format defined in Section IV-B.
4) **LLM Inference:** The prompt u_k is passed to the LLM, and the output y_k is parsed in a single pass using a regular expression to extract the first valid JSON object.
5) **Decision Mapping:** Depending on the control mode (guided or weighted), the final action a_k is computed as described in Section IV-A.
6) **Action Execution:** The selected action a_k is executed in the environment, transitioning the simulation to the next timestep.

From a systems perspective, the control architecture forms a standard closed loop in which sensors measure the environment, the controller computes control decisions, and actuators apply the resulting action. To guarantee feasible control at every timestep, invalid LLM outputs (e.g., malformed JSON or violations) trigger a fallback to PPO control.

V. Numerical Evaluation

In this section, we evaluate the proposed LLM-guided hybrid control framework for residential HP management. We consider a control interval of 5 minutes and an episode length of 288 timesteps, corresponding to a 24-hour operation cycle. We evaluate a hybrid control framework in which a LLM either selects (guided mode) or combines (weighted mode) a set of RL controllers to determine the final action. All experiments are conducted within the *LLECBuildingGym* [3] framework using the `LLEC-HeatPumpHouse-1R1C` environment. Each run records observations, actions, rewards, and LLM decisions, together with prompts and completions in JSON to ensure reproducibility. We first describe the experimental settings (V-A), then analyze the sensitivity of decoding hyperparameters (τ, p) (V-B), and finally evaluate additional commercial LLMs (V-C).

A. Experimental Settings

All experiments are run on a computer node equipped with 38 CPU cores and an NVIDIA A100-SXM4 GPU (MIG 4g/20 GB), using CUDA 12.4 and cuDNN 9.0.5.

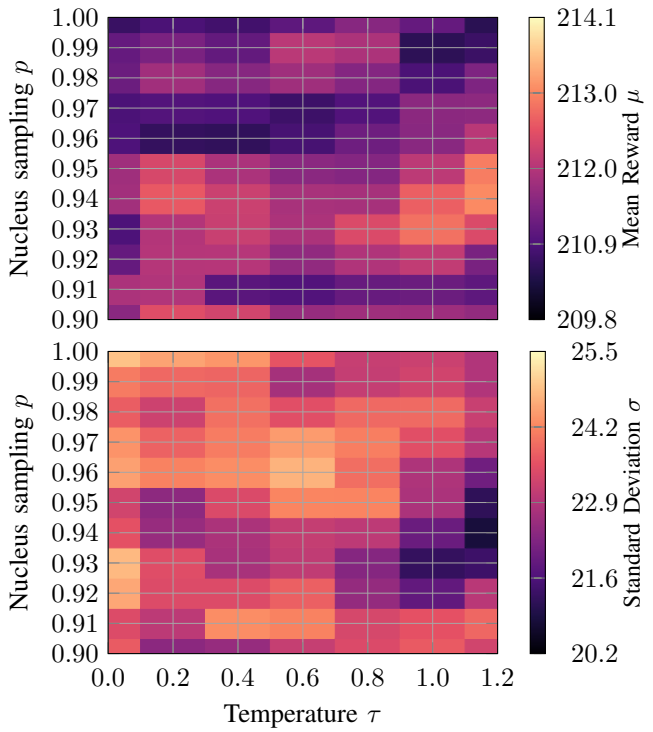

Fig. 2: Mean cumulative reward (top) and standard deviation (bottom) for the Mistral-7B model in guided mode.

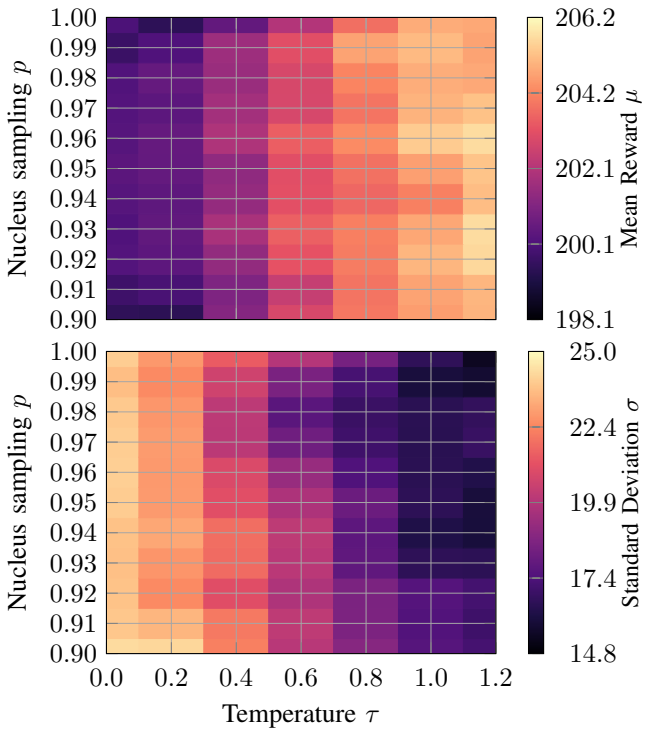

Fig. 3: Mean cumulative reward (top) and standard deviation (bottom) for the Llama-3.1-8B model in guided mode.

For local LLM experiments, Mistral-7B-Instruct-v0.3 (Mistral-7B) [20] is executed on MIG 4g/20 GB, while Meta-Llama-3.1-8B-Instruct (Llama-3.1-8B) [21] is executed using two MIG instances (4g/20 GB and 2g/10 GB). The DeepSeek-Chat [18] and GPT-4.1 [19] models are accessed via OpenAI-compatible REST APIs.[1]

All RL agents are trained using Stable-Baselines3 [30]. All standalone RL baselines are trained using a shared, fixed configuration in Table II without algorithm-specific hyperparameter tuning. All remaining parameters follow the SB3 v2.2.1 defaults to isolate the effect of LLM supervision.

B. Decoding Hyperparameter Sensitivity Analysis

To quantify the impact of the decoding parameters introduced in Section III-C, a two-dimensional sensitivity analysis is conducted across the temperature τ and nucleus sampling threshold p. Prior work [27], [31] indicates that moderate τ values combined with high p thresholds can mitigate degeneration while maintaining diversity in the sampled outputs. A total of 77 configurations are evaluated by sweeping $\tau \in \{0.0, 0.2, \ldots, 1.2\}$ and $p \in \{0.90, 0.91, \ldots, 1.00\}$ for Mistral-7B and Llama-3.1-8B in guided and weighted modes. For each (τ, p) configuration, five evaluation episodes are conducted with different random seeds. Figures 2–5 compare the mean μ and standard deviation σ of the cumulative episode reward to capture the sensitivity to τ and p.

[1]Versions: DeepSeek-Chat `2025-12-01`, GPT-4.1 `2025-04-14`.

Table III complements the reward analysis with output-validation statistics, showing the per-call rates of invalid JSON outputs (inv) and fallback activations (fb). Each control step allows at most $A = 2$ attempts (one initial call and one retry). We define the invalid-output rate per LLM request (call) and the fallback rate per control step as

$$r_{\text{inv}}(\tau, p) = \frac{N_{\text{inv}}(\tau, p)}{N_{\text{req}}(\tau, p)}, \qquad r_{\text{fb}}(\tau, p) = \frac{N_{\text{fb}}(\tau, p)}{N_{\text{steps}}(\tau, p)}, \quad (14)$$

where N_{inv} counts invalid JSON responses across all requests (up to two per step), and N_{fb} counts the number of steps in which both attempts fail, triggering the PPO fallback control.

TABLE II: RL training hyperparameters.

Parameter	**Value**
Policy type	Multi-Layer Perceptron Policy
Activation functions	`tanh` (PPO, A2C), `ReLU` (SAC, DDPG)
No. parallel envs	4
Learning rate α	3×10^{-4}
Discount factor γ	0.99
Total training steps	1×10^{6}
Rollout length	288
GAE parameter λ	0.95 (PPO), 1.0 (A2C)
Replay buffer size	1×10^{5}
Soft update coefficient	5×10^{-3}

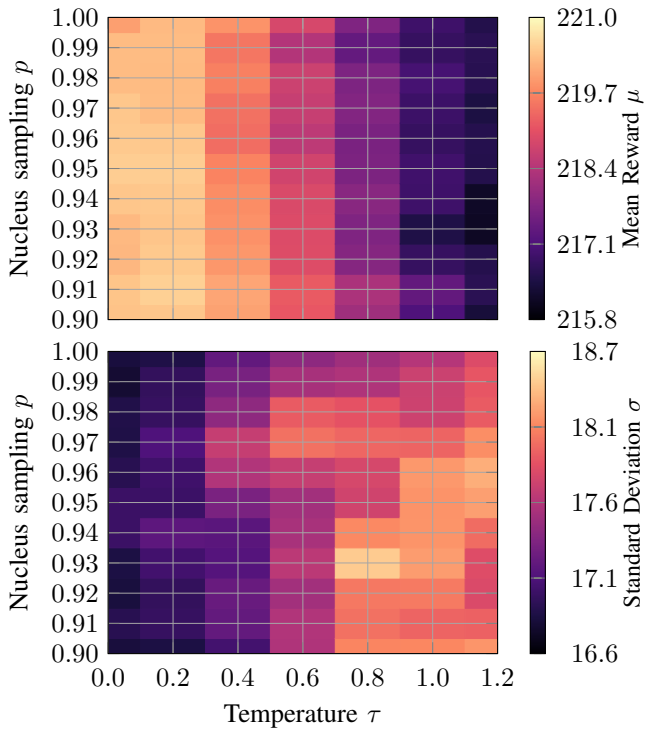

Fig. 4: Mean cumulative reward (top) and standard deviation (bottom) for the Mistral-7B model in weighted mode.

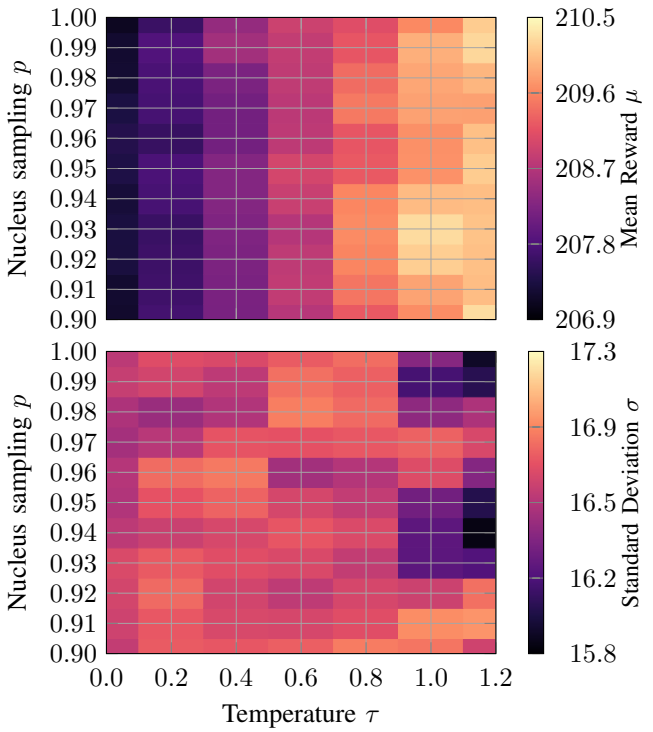

Fig. 5: Mean cumulative reward (top) and standard deviation (bottom) for the Llama-3.1-8B model in weighted mode.

1) Guided Mode: For Mistral-7B, Fig. 2 indicates weak sensitivity to (τ, p), with $\mu \in [209.8, 214.1]$ over the full grid. High-reward settings concentrate in $\tau \in [0.2, 0.4]$ and $p \in [0.90, 0.95]$ with moderate variability ($\sigma \approx 19.5$–21.4), indicating robust and reproducible guided decisions. For Llama-3.1-8B, the guided mode is more temperature-sensitive (see Fig. 3). For low temperatures ($\tau \leq 0.2$), rewards cluster around $\mu \approx 198$–200, and improve consistently for higher temperatures, reaching $\mu \geq 204$ for $\tau \geq 0.6$. The best mean reward is obtained at $(\tau, p) = (1.2, 0.96)$ with $(\mu, \sigma) = (206.2, 15.5)$ followed by $(1.0, 1.00)$ with $(\mu, \sigma) = (206.1, 16.3)$. Across both LLMs, Table III indicates that worst-case invalid-output and fallback rates are concentrated at high temperature and near-unconstrained nucleus sampling ($\tau = 1.2$ and $p \in \{0.99, 1.00\}$).

In summary, the guided mode with Mistral-7B achieves its best performance at moderate temperatures with moderately concentrated nucleus sampling, while Llama-3.1-8B benefits from higher temperatures.

2) Weighted Mode: For Mistral-7B, weighted mode performs best at low temperatures. Fig. 4 shows the mean cumulative reward and standard deviation over the (τ, p) grid. The maximum mean reward is obtained at $(\tau, p) = (0.2, 0.95)$ with $(\mu, \sigma) = (221.0, 17.0)$, closely followed by $(0.2, 0.97)$ with $(\mu, \sigma) = (220.9, 16.7)$. High-reward configurations concentrate over a wide range of p at $\tau \leq 0.4$, while increasing τ is coupled with a systematic decrease in performance, and achieves a value of $\mu = 216.1$ at $(\tau, p) = (1.2, 1.0)$. Output validity remains high across settings. The worst case in Table III occurs at $(\tau, p) = (1.2, 0.99)$ with 0.69% invalid outputs and a 0.28% fallback rate. For Llama-3.1-8B, weighted mode benefits from higher temperatures and peaks around $\tau \approx 1.0$. Fig. 5 shows the best performance at $(\tau, p) = (1.0, 0.94)$ with $(\mu, \sigma) = (210.5, 16.6)$, closely followed by $(1.0, 0.93)$ with $(210.4, 16.7)$. The output validity degrades towards the $p = 1.0$ corner. For $(\tau, p) = (1.2, 1.0)$, Llama-3.1-8B generates 5.97% invalid outputs and triggers the fallback

TABLE III: Top-10 worst-case (τ, p) configurations by invalid-JSON and fallback rates in guided and weighted modes.

			Guided (%)			**Weighted** (%)			
rk	τ	p	Mistral-7B		Llama-3.1-8B	Mistral-7B		Llama-3.1-8B	
			r_{inv}	r_{fb}	r_{inv}	r_{inv}	r_{fb}	r_{inv}	r_{fb}
1	1.20	1.00	1.81	0.00	4.65	0.28	0.28	5.97	9.31
2	1.20	0.99	1.39	0.00	2.22	0.69	0.21	5.42	7.85
3	1.20	0.98	2.15	0.07	0.83	0.28	0.14	3.12	5.49
4	0.60	0.93	0.00	0.00	0.00	0.00	0.00	0.00	8.47
5	0.60	0.95	0.00	0.00	0.00	0.00	0.00	0.00	8.12
6	0.60	0.96	0.00	0.00	0.00	0.00	0.07	0.00	7.92
7	1.20	0.96	1.39	0.00	0.42	0.07	0.00	2.15	5.56
8	0.60	0.91	0.00	0.00	0.00	0.00	0.00	0.00	7.64
9	0.60	0.92	0.00	0.00	0.00	0.00	0.00	0.00	7.43
10	0.60	0.97	0.00	0.00	0.00	0.00	0.07	0.00	7.29

r_{inv} (% per request) and r_{fb} (% per step) as defined in Eq. (14).

in 9.31% of cases, as summarized in Table III. Consequently, selecting $p \in [0.93, 0.98]$ provides a practical operating region for Llama-3.1-8B that avoids the $p \approx 1.0$ failure regime while retaining reward gains at high τ.

Overall, Mistral-7B is largely robust to decoding choices in weighted mode, with best performance at low τ. In contrast, Llama-3.1-8B prefers higher τ but requires moderately nucleus sampling to preserve valid outputs. Both LLMs remain reliable near $\tau \approx 0.8$ with $p \in [0.94, 0.97]$.

C. Additional LLM Performance Analysis

To validate generalization and stability, we select (τ=0.8, p=0.94) as a robust point across LLMs, based on the sensitivity analysis in Section V-B. Each controller is evaluated over 10 daily episodes (288 steps per day) with distinct random seeds (one per episode).

Table IV summarizes mean performance and 95% percentile bootstrap confidence intervals (CIs) for four LLMs (Mistral-7B, Llama-3.1-8B, DeepSeek-Chat, and GPT-4.1). It also presents results for four standalone RL baselines (PPO, SAC, DDPG, and A2C) under both guided and weighted control modes. Percentile bootstrap CIs are recommended for few-run RL [32]. In the guided mode, the GPT-4.1-based LLM-guided RL controller achieves up to a 5.1% higher mean cumulative reward (232.7) over the best standalone RL baseline. DeepSeek-Chat follows closely with $(\mu, \sigma) = (231.3, 17.0)$, outperforming Mistral-7B $(208.4, 23.6)$ and Llama-3.1-8B $(204.5, 18.3)$ by 11.0% and 13.1% in mean reward, respectively. Among standalone RL baselines, PPO performs best with $(\mu, \sigma) = (221.4, 18.5)$. SAC $(198.1, 27.8)$ and A2C $(196.3, 20.7)$ yield lower mean rewards. DDPG performs worst with $(182.7, 57.3)$, indicating substantially higher variability. In the weighted mode, where the LLM combines the actions of multiple RL agents, GPT-4.1 again achieves the best performance with $(\mu, \sigma) = (228.9, 18.6)$, followed by DeepSeek-Chat with $(224.9, 18.3)$. The open-source LLMs yield lower mean returns with $(216.0, 22.4)$ for Mistral-7B and $(206.4, 18.3)$ for Llama-3.1-8B. Overall, guided control slightly outperforms weighted control for the top-performing LLM (GPT-4.1: $\mu = 232.7$ vs. $\mu = 228.9$), and both modes consistently outperform the standalone RL baselines in mean cumulative reward.

To compare the best-performing LLM-guided controller against the best standalone RL baseline (PPO) under identical randomness, we compute the empirical mean paired difference in episode returns across matched seed–episode pairs as:

$$\bar{\Delta} = \frac{1}{N} \sum_{(s,e)\in\mathcal{P}} \left(R_{s,e}^{\mathrm{LLM}} - R_{s,e}^{\mathrm{PPO}}\right), \tag{15}$$

where $R_{s,e}^{\mathrm{LLM}}$ and $R_{s,e}^{\mathrm{PPO}}$ denote the final cumulative episode returns obtained under the same seed–episode pair (s, e), $\mathcal{P}$ is the set of matched pairs, and $N = |\mathcal{P}|$ (here $N = 10$). Using matched seed–episode pairs, GPT-4.1 (guided mode) yields $\bar{\Delta} = 11.39$ (95% paired bootstrap CI: $[5.39, 18.36]$), with $p_{\mathrm{perm}} = 0.0019$ from a two-sided paired sign-flip permutation test. Similarly, DeepSeek-Chat (guided mode) yields $\bar{\Delta} = 9.93$ (95% paired bootstrap CI: $[5.09, 15.33]$).

TABLE IV: Performance of LLMs (guided/weighted) and standalone RL baselines in the `LLEC-HeatPumpHouse-1R1C` environment.

Controller	Mode	Mean (μ)	CI Low*	CI High*	Std. (σ)
Mistral-7B	guided	208.4	194.8	222.5	23.6
Llama-3.1-8B	guided	204.5	194.0	215.7	18.3
DeepSeek-Chat	guided	231.3	221.3	241.2	17.0
GPT-4.1	guided	232.7	222.9	242.3	16.5
Mistral-7B	weighted	216.0	203.0	229.4	22.4
Llama-3.1-8B	weighted	206.4	196.1	217.5	18.3
DeepSeek-Chat	weighted	224.9	214.4	236.0	18.3
GPT-4.1	weighted	228.9	217.9	239.9	18.6
PPO	standalone	221.4	210.4	232.2	18.5
SAC	standalone	198.1	182.3	215.3	27.8
DDPG	standalone	182.7	146.7	213.1	57.3
A2C	standalone	196.3	183.7	208.3	20.7

*95% CIs. Fixed decoding parameters ($\tau = 0.8$, $p = 0.94$).

TABLE V: Runtime analysis for LLM-assisted controllers.

Controller	Inference latency (ms)		Cost
	Guided	Weighted	($/req.)
Mistral-7B	1292.45 ± 5.56	1341.77 ± 5.41	—
Llama-3.1-8B	1008.77 ± 1.49	1104.73 ± 4.92	—
DeepSeek-Chat	2813.51 ± 24.61	3265.01 ± 36.25	0.00015
GPT-4.1	1641.29 ± 45.07	1669.96 ± 28.90	0.00191
Standalone RL	14.03 ± 0.02		—

Table V quantifies the runtime overhead of LLM-assisted control in terms of per-step inference latency (ms) and summarizes per-request API costs for commercial models. The standalone RL controller requires 14.03 ± 0.02 milliseconds per step. LLM-assisted control increases decision time from milliseconds (ms) to seconds relative to standalone RL. A comparative runtime analysis across LLMs shows that the induced overhead is analogous in both guided and weighted modes. The maximum observed inference latency is below 3.3s. Consequently, this approach remains suitable for supervisory or high-level controller applications with minute-level update rates [33]. At a 5-minute update rate (288 decisions/day), the API costs are approximately US$0.043/day for DeepSeek-Chat and US$0.55/day for GPT-4.1, corresponding to US$0.00015 and US$0.00191 per request, respectively.

VI. Conclusion

This work presents a hybrid high-level control framework that leverages large language models (LLMs) as meta-policies that supervise reinforcement learning (RL) agents for heat pump control in smart buildings. Two operation modes are introduced, the guided mode for controller selection and the

weighted mode for action fusion, both communicating via stateless API calls.

Experiments with four LLMs demonstrate that LLM-guided RL control can outperform the best standalone RL agent by up to 5.1% in mean cumulative reward, while reducing the standard deviation by 10.8%. A two-dimensional decoding sensitivity analysis indicates that reward and output validity depend on (τ, p) and that a reliable operating region is obtained around $\tau \approx 0.8$ with $p \in [0.94, 0.97]$, where invalid outputs and fallback rates remain low. Furthermore, comparing the two meta-policy modes shows that the guided mode produces low-variance decisions, while the weighted mode achieves comparable performance by integrating complementary control actions. Commercial LLM performance is time-dependent due to model drift. Deployment further depends on external connectivity and usage-based pricing. In contrast, open-source LLMs and local inference stacks are improving rapidly and may close this gap. Local inference also improves data privacy by keeping all processing on-device. Moreover, the evaluation is limited to a single building model, and generalization across thermal parameterizations, building types, and tariff regimes requires further study.

Future research will extend the work to multiple building models and conduct robustness sweeps under forecast noise, extreme price events, and thermal-parameter variations. We will compare LLM-based supervision with learned meta-policies and hierarchical RL. Further directions include heterogeneous device portfolios, hardware-in-the-loop validation, multi-building coordination, and conversational interfaces for transparent decision-making.

VII. Acknowledgements

This work was supported in part by the Energy System Design (ESD) Project; in part by the Helmholtz Association's Initiative and Networking Fund through Helmholtz AI and under grant no. VH-NG-1727; and the HAICORE@KIT partition.

References

[1] Eurofound, "Decarbonisation of residential heating and cooling: The heat pump challenge," Luxembourg, 2024.
[2] T. Brudermueller *et al.*, "Estimation of energy efficiency of heat pumps in residential buildings using real operation data," *Nature Communications*, vol. 16, no. 1, p. 2834, 2025.
[3] G. Demirel *et al.*, "Advanced Deep Reinforcement Learning for Heat Pump Control in Residential Buildings," in *Proc. IEEE PES ISGT Europe*, 2025, pp. 1–5.
[4] I. Singh *et al.*, "ProgPrompt: Generating Situated Robot Task Plans using Large Language Models," in *Proc. of ICRA 2023*, 2023, pp. 11 523–11 530.
[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: A Bradford Book, 2018.
[6] O. Nachum *et al.*, "Data-Efficient Hierarchical Reinforcement Learning," in *NeurIPS*, 2018, pp. 3307–3317.
[7] W. X. Zhao *et al.*, "A Survey of Large Language Models," 2025.
[8] S. Minaee *et al.*, "Large Language Models: A Survey," 2025.
[9] M. Liu *et al.*, "Large Language Models for Building Energy Applications: Opportunities and Challenges," *Building Simulation*, vol. 18, no. 2, pp. 225–234, 2025.
[10] L. Zhang and Z. Chen, "Large language model-based interpretable machine learning control in building energy systems," *Energy and Buildings*, vol. 313, p. 114278, 2024.
[11] X. Ren *et al.*, "Can Large Language Model Agents Balance Energy Systems?" 2025.
[12] F. Michelon *et al.*, "Large Language Model Interface for Home Energy Management Systems," in *Proc. ACM e-Energy '25*, 2025, pp. 590–602.
[13] A. Sommer *et al.*, "Adaptive Self-Improvement for Smarter Energy Systems Using Agentic Policy Search," *SIGENERGY Energy Inform. Rev.*, vol. 5, no. 3, pp. 259–274, Dec. 2025.
[14] Y. Cheng *et al.*, "A Large Language Model for Advanced Power Dispatch," *Scientific Reports*, vol. 15, no. 1, p. 8925, 2025.
[15] M. Jia *et al.*, "Enhancing LLMs for Power System Simulations: A Feedback-Driven Multi-Agent Framework," *IEEE Transactions on Smart Grid*, vol. 16, no. 6, pp. 5556–5572, 2025.
[16] A. Bernadic *et al.*, "Large Language Models in Power Systems: Enhancing Control and Decision-Making," *International Journal of Innovative Solutions in Engineering*, vol. 1, pp. 10–17, 2025.
[17] J. Ruan *et al.*, "Applying Large Language Models to Power Systems: Potential Security Threats," *IEEE Transactions on Smart Grid*, vol. 15, no. 3, pp. 3333–3336, 2024.
[18] DeepSeek-AI *et al.*, "DeepSeek-V3.2: Pushing the Frontier of Open Large Language Models," 2025. [Online]. Available: https://arxiv.org/abs/2512.02556
[19] OpenAI *et al.*, "GPT-4 Technical Report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774
[20] A. Q. Jiang *et al.*, "Mistral 7B," 2023. [Online]. Available: https://arxiv.org/abs/2310.06825
[21] A. Grattafiori *et al.*, "The Llama 3 Herd of Models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783
[22] J. Schulman *et al.*, "Proximal Policy Optimization Algorithms," 2017.
[23] T. Haarnoja *et al.*, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," 2018.
[24] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2019.
[25] V. Mnih *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," 2016.
[26] D. H. Ackley *et al.*, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, no. 1, pp. 147–169, 1985.
[27] A. Holtzman *et al.*, "The Curious Case of Neural Text Degeneration," 2020. [Online]. Available: https://arxiv.org/abs/1904.09751
[28] Y. Lu *et al.*, "Learning to Generate Structured Output with Schema Reinforcement Learning," in *Proc. 63rd ACL*, 2025, pp. 4905–4918.
[29] C. Shorten *et al.*, "StructuredRAG: JSON Response Formatting with Large Language Models," 2024. [Online]. Available: https://arxiv.org/abs/2408.11061
[30] A. Raffin *et al.*, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.
[31] M. Nadeem *et al.*, "A Systematic Characterization of Sampling Algorithms for Open-ended Language Generation," in *Proc. 1st AACL and 10th IJCNLP*, 2020, pp. 334–346.
[32] R. Agarwal *et al.*, "Deep reinforcement learning at the edge of the statistical precipice," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
[33] Bosch Thermotechnology Corp., "Bosch IDS 2.0 (BOVA/BVA) Service Manual (Rev. 3)," 2021.