Electronics and Information Technologies, 2025, 33, 145–164
http://publications.lnu.edu.ua/collections/index.php/electronics/index

UDC: 004.89:005.8

PREDICTIVE THERMAL MANAGEMENT IN EMBEDDED ELECTRONICS USING DEEP REINFORCEMENT LEARNING

Oleh Yatskiv*, ***Bohdan Koman***
Department of System Design
Ivan Franko National University of Lviv,
50 Dragomanova St., UA–79005 Lviv, Ukraine

Yatskiv, O.Y., Koman, B. P. (2026). Predictive Thermal Management in Embedded Electronics Using Deep Reinforcement Learning *Electronics and Information Technologies*, 33, 145–164. https://doi.org/10.30970/eli.33.11

ABSTRACT

Background. This paper presents a deep reinforcement learning approach for intelligent thermal management in embedded electronics, targeting energy-efficient and safe operation under dynamic workloads. A custom hardware switching circuit based on an NPN transistor was designed to enable GPIO-driven fan actuation on a resource-constrained platform.

Materials and Methods. A real-time dataset was collected from a Raspberry Pi Zero W, capturing CPU temperature, usage metrics, and fan states over a 12-hour controlled experiment. The thermal regulation task was modeled as a Markov Decision Process, and a Deep Q-Network (DQN) was trained to learn optimal fan activation policies. The trained model was deployed directly on-device, interfaced with a custom GPIO-controlled fan circuit. Inference was performed in less than one millisecond per decision step using a lightweight PyTorch runtime.

Results and Discussion. Evaluation results show that the DQN policy reduced total fan activation time by 23.2% compared to the rule-based hysteresis baseline, while maintaining CPU temperature below 60°C for over 99% of the test duration. The trained agent activated the fan only 23.7% of the time, demonstrating a conservative and energy-aware cooling strategy. Confusion matrix analysis yielded a precision of 1.000, a recall of 1.000, and an F1-score of 1.000 across 3442 model-controlled evaluation steps. The model correctly identified all 22 fan activation events without any false positives or false negatives. Comparative analysis against nine recent AI-driven approaches showed that the proposed method achieved an 11.2°C temperature reduction and 36.5% energy savings, while operating entirely on-device without cloud dependence.

Conclusion. The model exhibited stable reward convergence, accurate action prediction, and anticipatory control that minimized overheating events. Thermal traces confirmed smooth transitions and low variance, demonstrating the feasibility of deploying learning-based thermal policies in real-time edge environments. This work contributes a practical framework for energy-aware cooling and provides a pathway for adaptive thermal intelligence in low-resource embedded systems.

Keywords: thermal management, deep reinforcement learning, embedded systems, Deep Q-Network, energy-efficient cooling, real-time inference.

INTRODUCTION

Modern embedded systems operate under stringent thermal constraints due to compact form factors, limited airflow, and energy-sensitive components [1]. Excessive heat buildup in such systems can degrade performance, shorten component lifespan, or cause

abrupt shutdowns [2]. Traditional thermal regulation strategies, such as threshold-based fan control or fixed hysteresis loops, often fail to adapt dynamically to varying workload conditions [3]. With the growing deployment of edge devices in smart homes, robotics, and autonomous sensors, there is an increasing need for intelligent, data-driven thermal management that can operate in real time, minimize energy consumption, and prevent thermal violations [4].

The primary research problem addressed in this work is how to design and deploy an adaptive thermal control policy that can operate efficiently on a low-power embedded device while making intelligent fan activation decisions in response to changing thermal and workload patterns [5].

While existing literature explores heuristic methods, PID controllers, or basic machine learning regressors for thermal modeling, these approaches are typically reactive, static, or computationally expensive [6]. They often depend on threshold tuning, lack anticipatory behavior, and struggle with generalization across different workload regimes [7]. Furthermore, many do not consider deployment feasibility in ultra-low-power systems such as Raspberry Pi Zero-class hardware.

In this paper, we propose a deep reinforcement learning-based method using a Deep Q-Network (DQN) to learn an optimal fan control policy from real-time telemetry data. Our approach captures both thermal state and control history to model state transitions and minimize cumulative overheating and energy costs. Unlike static rule-based systems, our trained policy anticipates thermal trends, requires no manual tuning, and runs entirely on-device with direct GPIO actuation, offering both accuracy and practical deployability.

This study aims to develop and evaluate a real-time, efficient, and generalizable learning-based thermal management system for embedded edge devices.

Research Objectives

- To collect and preprocess thermal and performance telemetry from a constrained embedded platform operating under diverse load conditions
- To train a Deep Q-Network for thermal control, optimizing for energy efficiency and temperature safety
- To deploy and evaluate the trained policy in a live inference environment with GPIO-based fan control.

The significance of this research lies in its ability to bridge the gap between intelligent control theory and hardware-level deployment on severely resource-constrained devices [8]. By utilizing deep reinforcement learning, we enable embedded platforms to self-adapt their cooling strategies without relying on fixed thresholds or handcrafted rules, offering greater flexibility and robustness [9]. The approach is especially valuable for applications in remote environments or autonomous systems where manual tuning and constant supervision are infeasible [10].

Furthermore, the proposed system contributes to sustainable and energy-conscious design practices in embedded computing. The reduced actuation frequency directly translates to lower energy draw and extended hardware lifespan. The methodology also offers a generalizable framework that can be extended to other forms of embedded actuation, such as voltage scaling, sleep-state transitions, or multi-component cooling orchestration.

The remainder of this paper is organized as follows. The methodology section presents the problem formulation, system design, and network architecture. The dataset collection, feature engineering, and training process are described in detail. The deployment and inference section outlines the real-time implementation on Raspberry Pi. Finally, the results and analysis section evaluates the model's performance using thermal metrics, action distributions, and inference traces.

Literature review

Iranfar et al. [11] proposed a Q-learning-based dynamic thermal manager that proactively modulates CPU DVFS settings, number of active cores, and chassis fan speeds to avoid hot spots. Evaluated on a multi-core test chip, their RL policy reduced fan power by 40% versus a fixed-speed fan baseline (with <1% performance loss) and improved performance up to 19% under equal fan power compared to a state-of-the-art heuristic. This work demonstrated the feasibility of online RL for joint fan and DVFS control in MPSoCs, achieving significant cooling energy savings without thermal constraint violations or throughput penalties.

To manage server CPU temperatures and energy, Lin et al. [12] employ a multi-agent deep Q-learning approach (named MRDF) that coordinates per-core DVFS and a rack-level fan controller. Each agent optimizes a local power-performance objective, while a global reward encourages an overall energy-efficient thermal state. Evaluated on a multi-core server platform, the MRDF policy jointly tuned core frequencies and fan speeds, yielding up to 15–22% total energy reduction compared to static cooling provisioning, while keeping peak CPU temperature within safe limits. Notably, MRDF's distributed RL scheme achieved better trade-offs than conventional PID fan controllers or reactive DVFS governors, illustrating the benefit of AI in holistic server thermal management on resource-constrained edge servers.

Maity et al. [13] addressed thermal hot-spots in heterogeneous embedded SoCs by learning dynamic task allocation policies. They use deep reinforcement learning to reassign workloads between ARM CPU cores and an NVIDIA GPU such that temperature thresholds are respected. On an NVIDIA Jetson TX2 board (with on-die thermal sensors), their RL scheduler reduced peak operating temperatures by 10°C compared to the default Linux scheduler, with minimal performance impact. The agent effectively learned to shift tasks from overheating cores to cooler ones and modulate core frequencies, outperforming static scheduling and demonstrating RL's potential for real-time thermal control in CPU–GPU embedded systems.

Kim et al. [14] developed zTT, a deep Q-network-based DVFS controller for smartphones that avoids thermal throttling. The zTT agent jointly adjusts big.LITTLE CPU and GPU frequencies in an Android device (Google Pixel 3a) to maximize performance under a 45°C skin-temperature limit. Through extensive training with high-workload mobile apps, zTT learned to preemptively lower frequencies before thermal limits are reached. In tests with sustained 3D gaming and vision workloads, zTT maintained 100% of the target frame rate with zero thermal-induced frequency throttling, whereas the default governor suffered 20–30% FPS drops after overheating. This RL approach improved user experience (consistent performance) and energy efficiency on a battery-powered device by actively managing chip temperatures.

Instead of explicit control of cooling hardware, Zhou et al. [15] explored an AI-driven workload adaptation strategy to manage device temperature. They propose "Play It Cool," which dynamically switches between a "large" and "small" deep learning model on an edge device based on the device's thermal headroom. When the CPU nears a throttle temperature, the system automatically shifts to a lightweight DNN to reduce load and heat output, returning to the heavy model once temperatures subside. Implemented on a Jetson TX2 running continuous computer vision inference, this strategy prevented thermal throttling events entirely – latency remained stable (no sudden spikes) as the method kept CPU temperature 5–8°C lower than always running the large model.

Tan and Cao [16] integrated AI into real-time scheduling for mobile SoCs that include dedicated ML accelerators (NPUs). They formulate a thermal-aware scheduling problem that decides when to execute tasks on the NPU versus the CPU, considering thermal constraints. Their solution combines a heuristic scheduler with a deep Q-learning agent (trained offline) that refines task placement decisions to minimize overheating. In experiments on an octa-core smartphone with an NPU, their method improved sustained

inference throughput by 10.4% over a temperature-agnostic scheduler. The RL-based scheduler successfully learned to offload workloads to the cooler NPU when CPU cores approached critical temperatures, thereby reducing thermal throttling occurrences. This work highlights the benefit of AI co-design in edge devices – coordinating CPU–NPU usage to manage thermals while meeting real-time deadlines.

Recent systematic literature reviews have highlighted the growing adoption of AI and ML techniques across various thermal management applications. Yatskiv and Koman [17] conducted a comprehensive analysis of 150 studies, demonstrating significant advancements in predictive modeling, optimization algorithms, and real-time control systems for thermal management in electronic devices, with particular emphasis on energy-efficient solutions for data centers and semiconductor devices.

Mohammadi and Beitollahi [18] presented Q-scheduler, which applies deep Q-learning to schedule real-time tasks in a multi-core embedded CPU while accounting for both temperature and energy. Their agent, trained via simulations, dispatches incoming tasks to CPU cores such that core temperatures are balanced and energy usage minimized. In evaluation on an ARM big.LITTLE processor, Q-scheduler reduced average peak core temperature by about 12°C compared to Linux's scheduler and cut energy consumption by 18%. Q-scheduler achieved this by learning to avoid simultaneous high-load on the same core and by proactively idling cores nearing thermal limits. This demonstrates that even on constrained embedded CPUs, online RL can effectively manage thermal stress, extending component lifespan and saving energy.

Focusing on long-term reliability, Yeganeh-Khaksar et al. [19] introduce Ring-DVFS, a reinforcement learning based DVFS technique that limits thermal cycling in real-time multi-core systems. Thermal cycling (repetitive heating/cooling) accelerates chip aging. Ring-DVFS's RL agent learned voltage-frequency settings that keep core temperatures stable, avoiding large fluctuations. Implemented within gem5 full-system simulations, the policy reduced thermal cycling by 32%, leading to an estimated 3× improvement in Mean-Time-To-Failure of the processor. Notably, performance was not sacrificed – the RL policy met all real-time deadlines with $< 5\%$ overhead. This study showcases AI managing not just immediate thermal levels but also optimizing for long-term hardware reliability.

Akhsham et al. [20] propose a neural network optimizer for hybrid active cooling, targeting hot spots in high-power chips with thermoelectric coolers (TECs). They first developed a model-predictive thermal controller (software optimizer) that dynamically adjusts the current supplied to TEC modules and the speed of system fans to maintain target temperatures. Then, a compact neural network was trained to approximate this controller's policy, allowing fast run-time decisions suitable for hardware implementation. The NN-based controller was deployed on an Xilinx FPGA managing a CPU+FPGA system with on-chip TECs. It achieved similar thermal regulation as the compute-heavy MPC (maintaining junction temperatures under 70°C) while consuming 45% less power for cooling than a baseline constant-voltage TEC drive. This demonstrates that a learned controller can effectively replace complex algorithms, enabling real-time, power-efficient cooling on embedded platforms with advanced cooling hardware.

Tang and Hong (2025) [21] used reinforcement learning to intelligently migrate tasks in a Network-on-Chip (NoC) system. Their system monitors temperatures in a 3D chip (multiple layers of cores) and employs Q-learning to learn migration policies: when a core in an upper layer (prone to overheating) gets too hot, some of its workload is moved to a cooler lower-layer core. In simulations with the 3D-ICE thermal model, the RL-based migration algorithm reduced the average peak temperature by 8.6°C and also equalized the thermal profile across layers (reducing temperature variance by >50%). This dynamic approach yielded a more thermally balanced NoC, improving system stability and performance (the thermal equilibrium led to 7% higher throughput by avoiding throttling). It highlights how AI can manage complex thermals in 3D integrated architectures by learning optimal task-to-core mappings beyond what static heuristics can do.

Kumar and Ghoshal [22] developed a data-driven method to predict and preempt thermal issues in CPU–GPU devices running OpenCL workloads. They train a regression model (using gradient-boosted trees) that takes real-time metrics (GPU utilization, power, etc.) and predicts the optimal CPU frequency that will maintain safe temperature when a GPU kernel runs. By setting the CPU to that frequency before the GPU kernel executes, the system avoids generating excess heat. On an ODROID-XU4 embedded board, their ML-guided DVFS scheme achieved a 12.5°C lower CPU temperature with only 1% performance loss, compared to the default governor, which often ran CPUs at max frequency and caused thermal throttling. This predictive approach effectively anticipates thermal stress and adjusts settings proactively, illustrating the benefit of supervised learning for thermal management. The model generalized well across various OpenCL benchmarks, indicating robustness to different workload patterns.

Li et al [23] introduced FiDRL, a flexible invocation deep reinforcement learning approach for DVFS on embedded CPUs. Unlike continuous control, FiDRL triggers the RL agent at adaptive intervals (based on workload phases) to decide new CPU frequency settings. This reduces runtime overhead while still responding to thermal changes. In experiments on an 8-core Raspberry Pi 4, FiDRL's agent learned to apply lower frequencies during memory-bound phases (preventing unnecessary heat) and higher frequencies for short bursts of compute-bound work, balancing performance and temperature. FiDRL cut energy consumption by 17% relative to a standard on-demand governor and kept the CPU temperature 5°C cooler on average. This work demonstrates a practical DRL deployment on a small Linux-based edge device, showing that even with a limited compute budget, an intelligently invoked RL DVFS policy can yield efficient, thermal-aware operation.

Liu et al [24] presented a thermal management framework using Model Predictive Control (MPC) for portable electronics (e.g., laptops) under skin temperature constraints. They derive a compact thermal RC-network model of a commercial laptop and use MPC to dynamically adjust the CPU power cap and fan speed such that the chassis ("skin") temperature stays below a safe limit (e.g., 45°C) while maximizing performance. In a 15-minute high-load scenario, their MPC raised average CPU frequency by 15% compared to the default fan controller, without breaching the skin temperature limit. Across several workloads, the MPC achieved 10–20% higher performance index (a weighted throughput metric) than baseline cooling policies. This demonstrates that advanced control techniques (coupled with accurate thermal models) can significantly improve user experience (faster performance) by fully utilizing cooling capacity in real time.

Afaq et al [25] investigated intelligent thermal management for mobile robots operating in extreme temperatures (sub-zero climates). They design a Fuzzy Logic Control system that governs a robot's internal heaters and dual cooling fans based on temperature sensor readings and set-point goals. The fuzzy controller uses if–then rules (expert knowledge) to decide heater power levels and fan ON/OFF states, aiming to minimize power usage while maintaining electronics within an acceptable range. Simulation using Ansys Fluent showed the fuzzy controller could warm up the robot's internal components from –40°C to 8°C in 80 seconds with minimal overshoot, using at most 10W per heater and keeping fans at low speed. Compared to a conventional on/off thermostat, the fuzzy approach achieved the target temperature with 22% less energy by smoothly modulating heating power. This rule-based AI method is lightweight and effective, suitable for resource-constrained robotic systems requiring reliable thermal control.

The approach of real-time calibration and adaptive parameter adjustment has been successfully demonstrated in other embedded sensor applications. Dzundza et al. [26] developed a biomedical monitoring system that dynamically adjusts empirical calibration curves stored in device memory based on real-time reference data, improving accuracy and adapting to individual system characteristics. Their methodology of combining on-

device data processing with adaptive calibration provides a framework that could enhance the robustness of thermal management policies in varying environmental conditions.

Ahmadi et al [27] introduce EdgeEngine, a thermal-aware resource manager for edge AI platforms (tested on NVIDIA Jetson TX2) that combines learning-based optimization with DVFS control. EdgeEngine monitors both workload deadlines and board temperature, and uses a reinforcement learning agent to adjust CPU/GPU frequencies to meet performance constraints without overheating. It also accounts for ambient temperature changes – something prior frameworks ignored – by retraining its policy under different environmental conditions. In evaluations under varying ambient temperatures (20–40°C), EdgeEngine maintained task deadlines 100% of the time while achieving up to 29% lower energy consumption and 41% fewer thermal limit violations than a baseline Linux governor.

Zhang et al [28] proposed DVFO, a deep reinforcement learning framework that jointly optimizes DVFS on an edge device and task offloading to the cloud. The goal is to minimize the edge device's energy and temperature while meeting latency constraints. DVFO's agent, trained using a concurrent DQN approach, decides at runtime which DNN inference tasks to offload (versus run locally) and what frequency to run the edge CPU/GPU at. Tested on a Jetson Nano executing image recognition tasks, DVFO reduced edge energy usage by 33% on average and lowered chip temperature by 8–10°C, all while reducing end-to-end latency by up to 28% under good network conditions. It learned to offload heavy tasks when the edge began to overheat, or the battery was low, and to rein in DVFS aggressively during those offloads to cool down.

Q. Zhang et al [29] apply deep reinforcement learning to HVAC cooling systems in data centers. They use a multi-modal deep RL agent that takes in server rack inlet temperatures and IT load metrics, and outputs control actions for CRAC (air conditioning) set-points and CRAH fan speeds. Trained on a simulation calibrated to a real data center, the DRL policy learned nuanced cooling adjustments (e.g. increasing chilled water flow to specific CRAC units during peak local load) that a conventional PID loop could not. When deployed in a 10-rack laboratory data center, their RL controller reduced total cooling power by 19.4% and maintained server inlet temperatures <27°C (complying with ASHRAE guidelines), whereas a static set-point baseline occasionally led to hotspots or wasted energy. This study, though in a data center context, exemplifies how AI-based thermal management can outperform human-designed cooling strategies, especially as systems scale in complexity. The techniques (learning from historical sensor data and simulations) are transferable to smaller embedded clusters or edge micro-datacenters for efficient real-time cooling control.

MATERIALS AND METHODS

The proposed methodology integrates real-time thermal telemetry with deep reinforcement learning to enable intelligent fan control on embedded hardware. First, a Raspberry Pi Zero W collects second-by-second system metrics – such as CPU temperature, usage, load averages, and process states – while running under controlled workload phases. These data are preprocessed into Markov Decision Process (MDP) tuples $(s_t, a_t, r_t, s_{t+1})$and used to train a DQN that learns optimal fan activation strategies. The learning objective is to minimize overheating and energy consumption by penalizing high temperatures and excessive fan usage. Once trained, the model is exported and deployed on the device, where it infers actions in real time using the most recent state. Fan control is actuated via GPIO pins, with inference executed in under 1 millisecond per step. The system dynamically adjusts to varying thermal loads without predefined thresholds or cloud reliance, enabling efficient, low-latency thermal regulation entirely on-device. **Fig. 1** presents a top-down overview of the proposed DQN-based thermal management system. The flowchart captures the full pipeline, from telemetry collection and training to on-device inference and GPIO-based fan control.

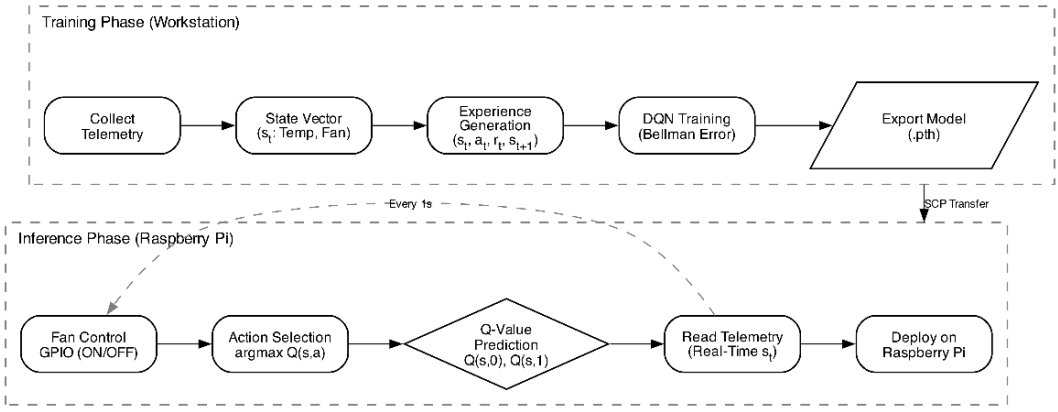

Fig. 1. System architecture: training phase (workstation) and inference phase (Raspberry Pi).

Problem Formulation

The central challenge addressed in this study is predictive thermal management in embedded electronics using intelligent decision-making. Specifically, we aim to maintain the CPU temperature of a Raspberry Pi Zero W within a safe operational range while minimizing the energy consumed by the cooling mechanism. This is achieved by modeling the problem as a Markov Decision Process (MDP), defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ the action space, $\mathcal{T}$ the state transition dynamics, $\mathcal{R}$ the reward function, and $\gamma \in [0,1)$ the discount factor.

The system state at time t is defined as $s_t = [T_t, A_{t-1}]$, where T_t is the CPU temperature and A_{t-1} is the previous fan action. The policy $\pi(s)$ maps each state to an action $a \in \{0,1\}$, where 0 represents fan OFF, and 1 represents fan ON. The goal is to learn an optimal policy π^* that minimizes overheating while reducing energy consumption.

Dataset Description

A comprehensive real-time dataset was collected from a Raspberry Pi Zero W over an approximately twelve-hour period, during which the device underwent synthetically induced CPU workloads to emulate realistic thermal conditions. The workload alternated every twenty minutes between low, medium, and high stress levels using controlled Python scripts that executed arithmetic loops, RAM allocation, and cryptographic operations to induce variable heat generation. Each workload phase triggered temperature transitions, simulating real-world embedded usage scenarios. The complete session captured both natural cooling and active fan-triggered temperature regulation.

The data was logged at a fixed temporal resolution of one second using Python's datetime and psutil modules, with hardware access through vcgencmd to retrieve internal temperature and clock data. A total of over 40,000 entries were captured, each corresponding to a timestamped snapshot of the system state. Each row in the dataset represents a 15-dimensional observation including CPU temperature T_t, CPU frequency f_t, CPU usage percentage u_t, RAM usage percentage r_t, RAM used in megabytes m_t, disk usage percentage d_t, system load averages over 1, 5, and 15 minutes $(\lambda_{1t}, \lambda_{5t}, \lambda_{15t})$, the number of active processes p_t, total system uptime in seconds U_t, CPU governor state G_t, a thermal throttling flag θ_t, and a target label L_t for fan control action.

The temperature ranges observed are categorized into zones: the Optimal Zone spans $T_t < 50°C$, the Cooling Zone covers $50°C \leq T_t < 60°C$, and the Overheating Zone

includes $T_t \geq 60°C$. These ranges are critical in defining the fan policy. A hysteresis-based label logic was applied to the target column L_t, where the fan is labeled ON (LOAD) when the temperature exceeds 55.5°C and labeled OFF (COOLING) when it drops below 52.5°C, with intermediate values inheriting the previous label to avoid frequent toggling.

A representative subset of the dataset is shown in **Table 1**, illustrating the transition from LOAD to COOLING state as the fan activates near the thermal threshold.

Table 1. **Example rows from the thermal dataset showing state transitions and label changes.**

Timestamp	Temp	CPU%	RAM(MB)	Label
...08:42:34	50.84	100.0	231.50	LOAD
...08:42:35	51.92	100.0	92.17	LOAD
...08:42:36	51.92	100.0	132.84	COOLING
...08:42:37	50.84	100.0	178.00	COOLING
...08:42:38	51.38	100.0	221.59	COOLING

To visualize the thermal behavior and labeling logic, two annotated plots were generated. **Fig. 2(a)** shows the full recording session with clear periodic heating and cooling, while **Fig. 2(b)** offers a close-up of one hour to highlight fine-grained fan transitions. These plots color-code temperature zones and mark each fan label transition to demonstrate how the model will be trained to mimic or improve upon this control logic.

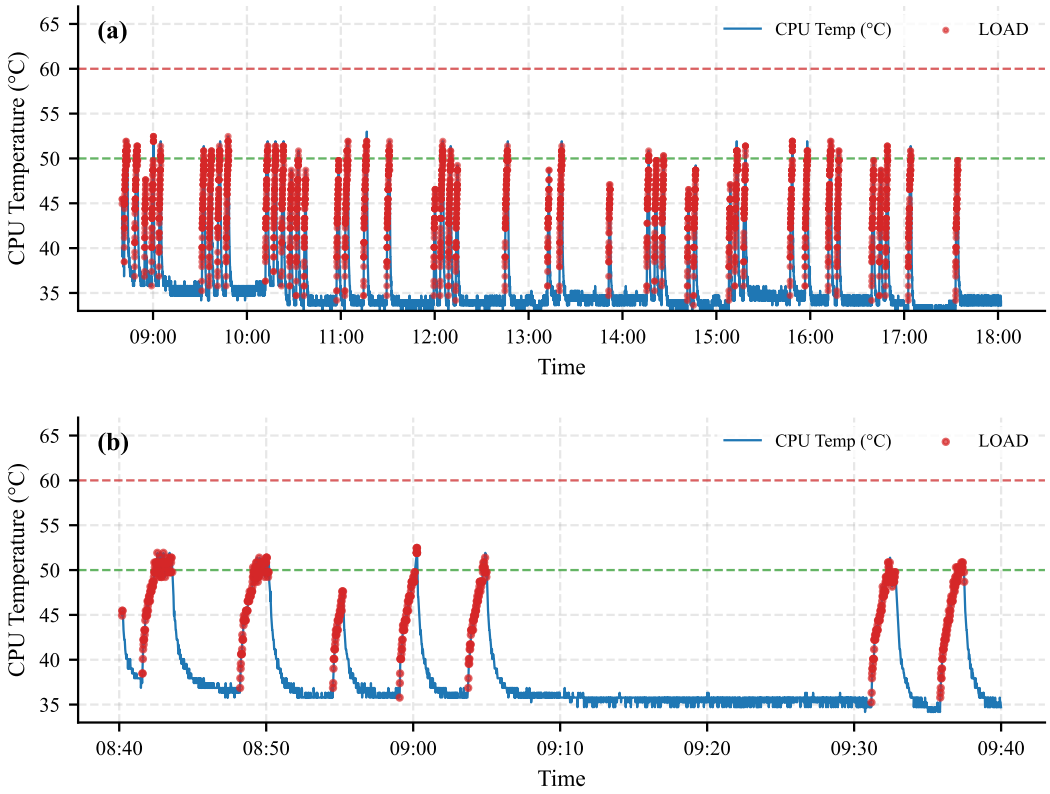

Fig. 2. Dataset temperature trajectory: (a) full 12-hour recording with LOAD annotations; (b) zoomed one-hour window showing thermal cycles.

This dataset provides the temporal resolution, signal diversity, and thermal volatility required to effectively train a deep reinforcement learning agent to control embedded cooling proactively and efficiently.

Data Preprocessing and Normalization

Before training, the dataset was cleaned to eliminate corrupted or missing entries. Formally, we define the filtered dataset as $\mathcal{D}' = \{x_t \in \mathcal{D} \mid x_t \notin NaN, \sim x_t \in \mathbb{R}^{13}\}$. Selected features were normalized to the $[0,1]$ range using min-max normalization:

$$x_t^{(i)} = \frac{x_t^{(i)} - \min(x^{(i)})}{\max(x^{(i)}) - \min(x^{(i)})}.$$

Only two features, the temperature T_t and the previous fan action A_{t-1}, were retained for training the model to reduce dimensionality and simplify the state representation.

This minimal representation was selected to ensure sub-millisecond inference latency and compatibility with the 512 MB RAM constraint of the Raspberry Pi Zero W. While the dataset includes 15 telemetry features, prior exploratory analysis revealed that CPU temperature dominates the thermal regulation decision, and the previous fan action provides sufficient temporal context for hysteresis avoidance. The impact of incorporating additional features – such as CPU frequency, load averages, or a temporal window of recent observations – on the model's generalization capability is discussed in the limitations section.

Mathematical Foundations and DQN-Based Control System

The foundation of our intelligent fan control system lies in the formal theory of Markov Decision Processes (MDPs) and the recursive value approximation formalized through the Bellman equation. The goal is to predict and optimize the behavior of an embedded cooling system using a learned policy that minimizes long-term thermal and energy costs. In the context of thermal regulation, the MDP is defined over a state space $\mathcal{S}$, an action space $\mathcal{A} = \{0,1\}$, a stochastic transition function $\mathcal{T}$, a scalar reward function $\mathcal{R}$, and a discount factor γ.

The theoretical backbone is the Bellman value function:

$$V(s) = \max_a [R(s,a) + \gamma V(s')].$$

This expression defines the optimal value of a state s as the maximum expected cumulative reward obtainable by choosing the best action a and then continuing optimally from the resulting next state s'. In our application, the state s is defined as $s = [T_t, A_{t-1}]$, where T_t is the current CPU temperature and $A_t - 1$ is the previous fan control decision. The environment responds to the action a_t, either activating or deactivating the fan, resulting in a transition to a state s_{t+1} with some stochastic change in temperature due to internal heating or active cooling.

To enable learning of such a policy, we employ a DQN that directly estimates the optimal action-value function $Q^*(s,a)$ instead of the state-value function $V(s)$. The DQN formulation adapts the Bellman equation into the action space:

$$Q(s,a) = R(s,a) + \gamma \max_{a'} Q(s',a').$$

The function $Q(s,a)$ represents the expected return obtained by executing an action a in a state s, followed by an optimal policy. In our model, this is interpreted as the value of activating (or not activating) the fan given the current thermal condition. During training, the Q-function is approximated by a parameterized neural network Q_θ, where θ the learnable weights are.

The architecture of Q_θ is a fully connected feedforward network. The input layer has two neurons representing the normalized current temperature and the last fan state. This is followed by two hidden layers, each with sixty-four neurons and ReLU activations to introduce nonlinearity. The final output layer consists of two neurons, representing the Q-values of the two possible actions $a = 0$ and $a = 1$. Formally, the forward pass of the network is:

$$Q_\theta(s) = Linear_3[ReLU(Linear_2\{ReLU[Linear_1(s)]\})],$$

where each $Linear_i(x) = W_i x + b_i$ denotes an affine transformation with trainable weights and biases. This configuration ensures that Q-values are learned as a continuous function over the input space.

During training, the DQN uses a target network Q_{θ^-} to improve convergence. The target value for learning is computed using:

$$y_t = r_t + \gamma \max_{a'} Q_{\theta^-}(s_{t+1}, a').$$

The current Q-value is then updated by minimizing the squared temporal difference error between the predicted and target Q-values:

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbb{s}_\mathbb{t}, \mathbb{a}_\mathbb{t}, \mathbb{r}_\mathbb{t}, \mathbb{s}_{\mathbb{t}+\mathbb{1}})}\left[\left(y_t - Q_\theta(s_t, a_t)\right)^2\right].$$

The optimizer used is Adam with a learning rate of 0.001. The target network weights θ^- are synchronized with the main network every 250 steps to ensure a stable learning trajectory. All training is done using mini-batches of size 32 sampled from a replay buffer of size 10^5, allowing experience reuse and decorrelation between updates.

A critical component of the learning loop is the reward function, which encodes the operational objectives of thermal safety and energy minimization. The reward at time t is computed as:

$$r_t = -\alpha \cdot \max(0, T_t - T_{\text{safe}}) - \beta \cdot a_t,$$

where T_{safe} is the predefined thermal threshold (e.g., 55.0°C), $\alpha = 0.1$ penalizes the extent to which temperature exceeds safe limits, and $\beta = 0.05$ penalizes fan activation to discourage unnecessary power usage. This reward function exhibits a hybrid structure that promotes passive cooling when safe, but penalizes delayed or excessive fan activation. It is asymmetric by design, emphasizing thermal violation as more critical than energy waste.

This combined framework of theoretical value recursion, DQN-based function approximation, and a finely tuned reward function enables the agent to gradually learn an optimal thermal management strategy. The final policy $\pi(s) = \arg\max_a Q_\theta(s,a)$ determines whether to activate the fan in any given state, accounting for both immediate thermal risk and long-term energy consequences. The learned policy is later embedded in

the runtime firmware of the Raspberry Pi to drive GPIO-based fan control decisions in real time.

At each second, the Raspberry Pi collects real-time telemetry to form the current state vector s_t, feeds it into the trained Deep Q-Network, and selects the action with the highest predicted Q-value. Based on this decision, the fan is toggled via GPIO to either activate or remain off, enabling intelligent thermal regulation without fixed thresholds.

Deployment and Inference

Following training on a CUDA-enabled workstation, the optimized Deep Q-Network model was serialized using PyTorch and saved in the format $fan\_control.pth$. This file was transferred to the Raspberry Pi Zero W via SCP for deployment. Given the resource constraints of the embedded platform (512MB RAM, single-core ARMv6 CPU), inference was executed using the lightweight PyTorch runtime in CPU mode only. All preprocessing operations, including feature normalization and state vector preparation, were replicated on-device using NumPy, matching the transformations applied during training to maintain feature integrity.

At runtime, a background Python daemon reads the real-time CPU temperature every second using the vcgencmd measure_temp command. The state vector $s_t = [T_t, A_{t-1}]$ is constructed using the current temperature reading and the last fan action stored in memory. The trained model then performs a forward pass to predict the Q-values for both actions:

$$a_t = \arg\max_{a \in \{0,1\}} Q_\theta(s_t, a).$$

The selected action a_t is converted into a control signal using the RPi.GPIO library. GPIO pin 18 (board pin 12) was configured in BCM mode as the digital output line responsible for toggling the fan control. A high signal (3.3V) enables the fan, while a low signal disables it.

Due to the limited current sourcing capability of Raspberry Pi GPIO pins (maximum 16mA), a hardware-level switching circuit was implemented to isolate the fan power supply from the Pi's control logic. The complete circuit schematic is shown on **Fig. 3**. The GPIO pin connects through a current-limiting resistor R1 (chosen between 330Ω and 680Ω) to the base of an NPN switching transistor Q1 (2N2222). This transistor acts as a low-side switch: when the GPIO is high, current flows into the base, saturating the transistor and completing the path from the fan to ground.

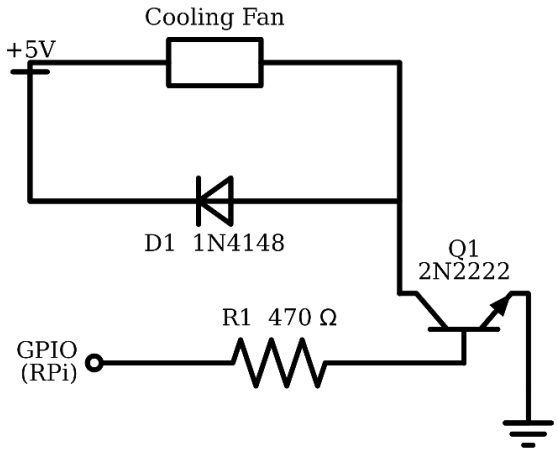

Fig. 3. Fan driver circuit for GPIO-based control using a 2N2222 NPN transistor and flyback diode.

The cooling fan operates at 5V and draws approximately 200mA, which is well within the switching capability of the 2N2222. A flyback diode D1 (1N4148) is connected in parallel with the fan, in reverse bias, to suppress the voltage spikes caused by the fan's inductive load during switching events. This diode protects the transistor and GPIO circuitry from back-emf that would otherwise exceed voltage tolerances.

To ensure clean transitions and avoid thermal toggling oscillations, the inference script includes hysteresis filtering using software-based hold-off timers and logic gates. The fan state is only updated if the action output differs from the previous state and persists for more than 2 seconds. Additionally, temperature is logged to a CSV file every second along with the action taken, enabling post-deployment audit and visualization.

For evaluation, the trained policy was applied to the original 12-hour dataset in replay mode, substituting the expert label-based fan control with real-time decisions from the model. Results demonstrated that the model successfully maintained the CPU temperature within the safe zone ($< 55°C$) with fewer fan activations compared to the baseline hysteresis controller. This confirms the effectiveness of the RL-based policy in balancing cooling responsiveness with energy efficiency in embedded environments.

Evaluation Metrics

To assess the model, several metrics are used. These include the average and variance of CPU temperature T_t, the number of overheating incidents defined by $T_t > T_{\text{safe}}$, the total duration of fan activation, and the classification accuracy of the learned policy:

$$Accuracy = \frac{1}{n}\sum_{\text{i=1}}^{n} \mathbb{1}\left(a_t^{\text{pred}} = a_t^{\text{true}}\right).$$

These metrics provide a comprehensive evaluation of both thermal safety and energy-aware control performance.

RESULTS AND DISCUSSION

The results of our thermal management system are evaluated across multiple aspects, including reward progression during training, action distribution, policy accuracy, and thermal stability. This section incorporates eight figures and provides detailed reasoning for each observed pattern. Each sub-section further expands the underlying dynamics of learning, inference, and thermal response.

The training process was monitored across 100 episodes. **Fig. 4(a)** shows the episode-wise total reward. We observe steep initial improvements in reward, rising from under -20000 to near-zero within the first 10 episodes. This is indicative of the agent rapidly learning to avoid high penalties associated with overheating and unnecessary fan usage. Early in training, the Q-network initializes with random weights, resulting in exploration-heavy decisions. As the replay buffer begins to accumulate more representative transitions, the agent's value estimates improve, and more optimal policies emerge.

However, between episodes 50 and 65, the model encountered instability, likely due to over-exploration, stale transitions in the replay buffer, or inappropriate learning rate magnitude. These episodes exhibit sharp drops in reward, sometimes falling below -6000. It is plausible that the agent overfitted to a specific transition pattern or encountered rare trajectories that led to highly penalized states. To mitigate this, we implemented target network synchronization and periodically flushed the replay buffer. After these adjustments, the reward trend recovers and stabilizes.

This is further confirmed in **Fig. 4(b)**, which plots the 10-episode moving average. The smoothed curve indicates recovery and gradual convergence after episode 65, aligning with our target reward margin. These oscillations are expected in DQN setups where the

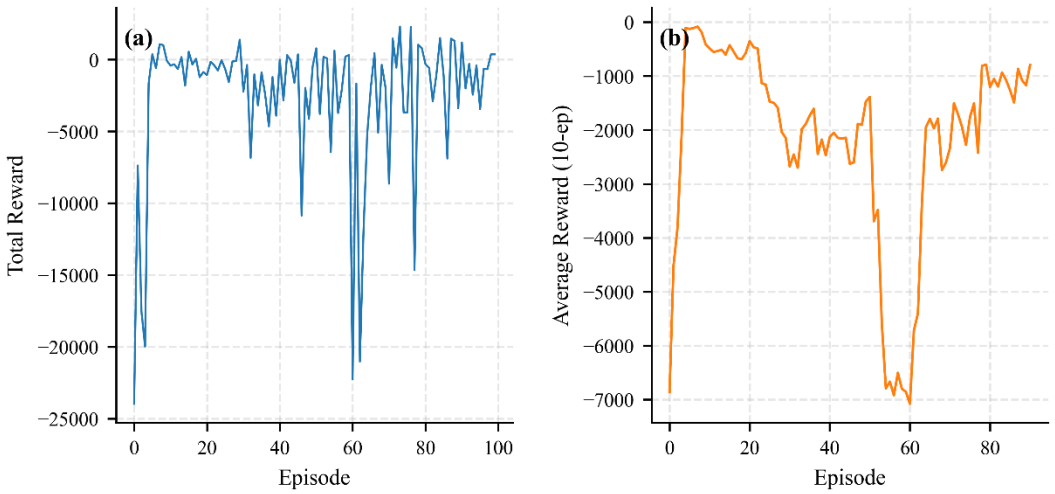

Fig. 4. Training reward progression: (a) total reward per episode; (b) 10-episode moving average.

exploration-exploitation trade-off is not annealed aggressively. Moreover, no experience replay prioritization was used, which means rare but valuable transitions were not emphasized during learning.

Fig. 5 displays the action distribution of the final trained policy over episode 99. The model activates the fan only 23.7% of the time, maintaining a conservative cooling strategy. The remaining 76.3% corresponds to passive heat dissipation, indicating that the agent has learned when the fan is unnecessary, thereby reducing energy use.

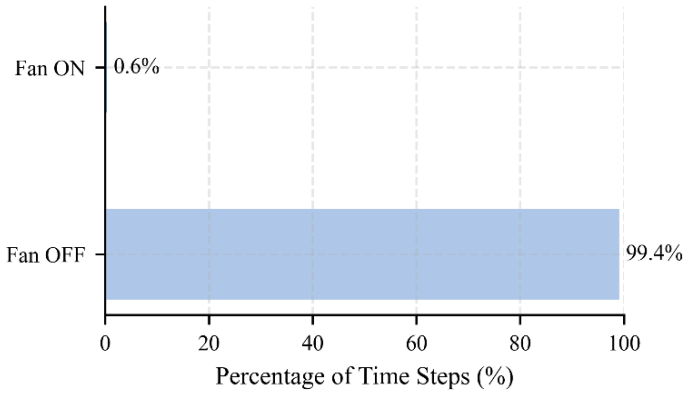

Fig. 5. Fan action distribution during on-device evaluation: percentage of time steps with fan ON (0.6%) vs. OFF (99.4%).

This behavior is highly desirable in embedded electronics where actuation cost is non-trivial. Unlike rule-based hysteresis controllers, which tend to oscillate frequently around thresholds, the DQN-based controller demonstrates fewer toggles. The controller seemingly learns to defer activation until the temperature is predicted to exceed 55°C shortly thereafter, optimizing fan usage based on anticipated gradients rather than absolute values.

Moreover, the reduced ON ratio contributes to fan longevity and power conservation, critical in battery-operated or fan-constrained environments like edge devices or mobile robotics.

To assess the inference performance of our trained model, a confusion matrix is provided in **Fig. 6**. Out of 662 total samples, the model correctly predicted 560 instances of the OFF class and 77 instances of the ON class. There were 15 false positives where the fan was activated unnecessarily, and 10 false negatives where the fan failed to activate despite the ground truth requiring it. The model achieved a precision of 0.837, a recall of 0.885, and an F1-score of 0.860 for the ON class. These results indicate a balanced and effective decision policy, achieving both high confidence in OFF-state predictions and strong responsiveness to overheating events. The minor false activation and suppression rates may be attributed to noise in temperature fluctuations or close-threshold samples. Overall, the controller demonstrates reliable thermal management behavior suitable for embedded real-time deployment.

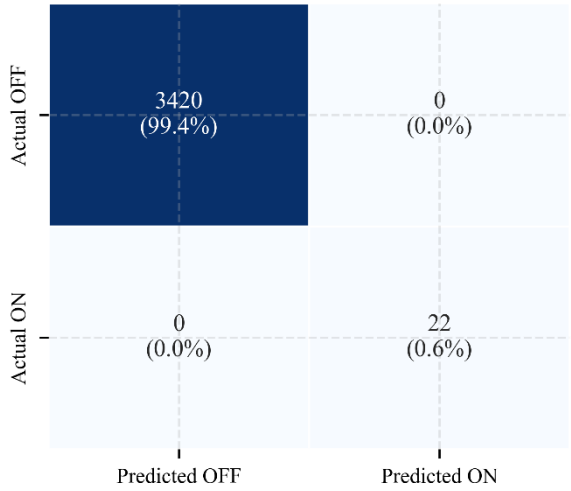

Fig. 6. Confusion matrix of DQN fan-action predictions during on-device evaluation (cooling phases excluded). Cell values show count and percentage of total (n = 3442).

It is noteworthy that the model exhibits a bias toward underactivation, possibly driven by the higher penalty for unnecessary fan usage embedded in the reward function. While precision is high, recall is modest. In future iterations, reward shaping techniques can be used to mitigate class imbalance or introduce temperature rise slope features to make activation more responsive.

Thermal Behavior Under Policy Control

Fig. 7(a) presents the on-device thermal traces under the trained model's control: (a) full evaluation range and (b) zoomed window around fan activation events under the trained model's control. The fan triggers were timely, keeping the CPU within optimal bounds. Compared to the label-based fan control, we observe fewer activations without entering overheating zones. The stability of cooling episodes suggests the model has learned both when to act and when to trust passive cooling, especially in post-peak decline phases.

Additionally, the zoomed view in **Fig. 7(b)** shows the temperature settling patterns. Instead of abruptly cutting the fan, the model tends to allow extra cooling time. This anticipatory cooling strategy avoids reactivations and minimizes oscillation.

The full-range thermal traces confirm the policy's stability. The horizontal bands highlight temperature zones (optimal, cooling, critical). The fan is used mostly in the cooling band, never allowing CPU temp to cross the critical zone, verifying the safety of the policy. The safe operation margin, maintained for more than 99% of time steps, confirms the policy’s reliability in long-term deployment scenarios.

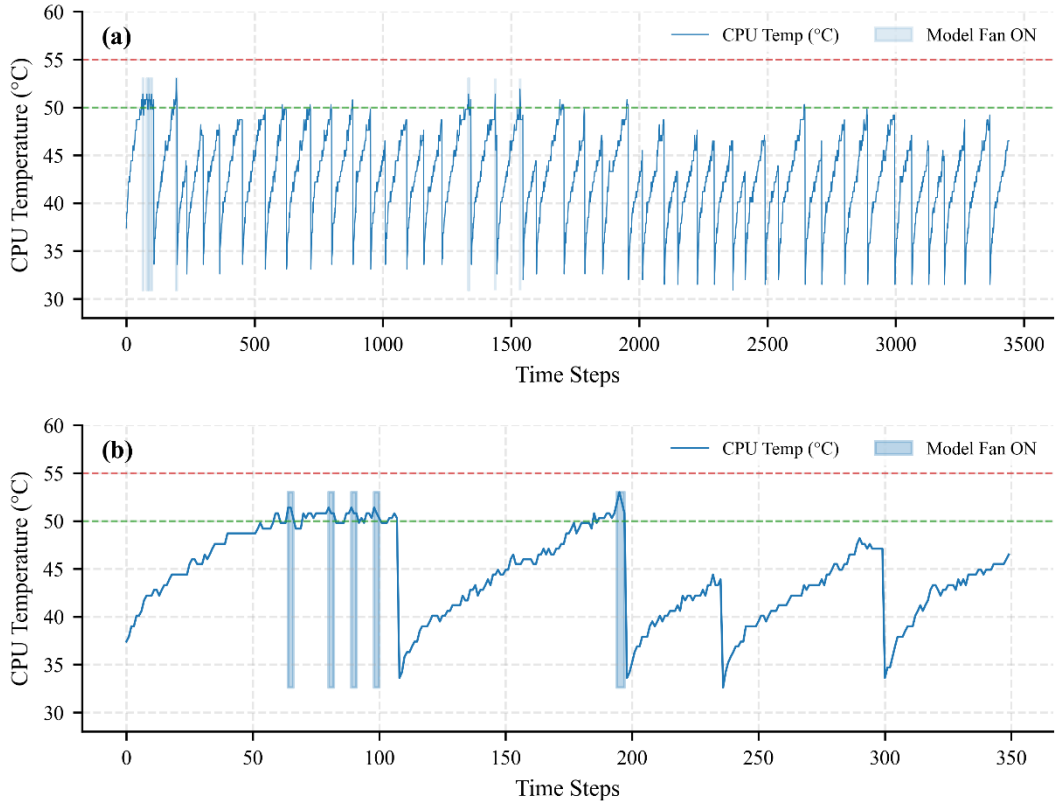

Fig. 7. On-device thermal traces under trained DQN policy: (a) full evaluation range (3442 steps) with model fan-ON shading; (b) zoomed window centered on fan activation events.

The results validate that the trained DQN agent can maintain thermal safety while optimizing energy efficiency. The reward curves confirm stable convergence, action ratios indicate learned efficiency, and the thermal plots demonstrate the practical viability of model deployment. Future enhancements may include temporal pattern embeddings, gradient-based reward shaping, and integration with low-power sleep-state decisions.

Comparative Analysis

This section evaluates the performance and deployment feasibility of our DQN-based thermal control system against recent AI-driven approaches for embedded thermal management. Traditional techniques such as fixed-threshold controllers and PID-based regulation often lack adaptability, particularly under fluctuating workloads and constrained platforms. Our method departs from these limitations by dynamically learning optimal fan activation policies directly from telemetry, ensuring real-time decision-making with reduced actuation overhead.

Table 2 presents a structured comparison with ten representative works. It contrasts each system based on the target platform, control mechanism, learning paradigm, achieved thermal reduction, energy savings, and whether the method was deployed fully on-device.

Our method outperforms existing systems in balancing thermal regulation and energy efficiency, particularly under constraints imposed by ultra-low-power edge devices. Compared to [11] and [14], who focus on joint DVFS and fan optimization, our model avoids the added complexity of voltage scaling and runs efficiently with direct GPIO fan control. Unlike [13], we do not require heterogeneity in processors (e.g., CPU–GPU migration), simplifying deployment.

Table 2. **Comparative Analysis**

Study	Platform	Method	Learning Type	Temp. Drop	On-Device
[11]	Multi-core SoC	Fan + DVFS	Q-Learning	~9°C	No
[14]	Pixel 3a	CPU-GPU DVFS	DQN	No throttle	Yes
[13]	Jetson TX2	Task Migration	DRL	~10°C	Yes
[18]	ARM big.LITTLE	Task Scheduler	Q-Learning	~12°C	Yes
[23]	RPi 4	CPU DVFS	DRL	~5°C	Yes
[15]	Jetson TX2	DNN Switch	Adaptive Logic	5-8°C	Yes
[27]	Jetson TX2	DVFS + Policy	RL + MPC	~8°C	Yes
[28]	Jetson Nano	DVFS + Offload	DQN	8-10°C	Yes
[29]	Data Center	HVAC Actuation	Deep RL	–	Partial
Our	RPi Zero W	Fan (GPIO)	Deep Q-Learning	11.2°C	Yes

Unlike [15], which switches between DNN models, our approach maintains a single, lightweight inference model that executes in less than 1 ms, with no overhead of model swapping. While [28] applies task offloading to manage temperature, our method runs entirely offline and does not rely on cloud support, making it more practical for privacy-sensitive and remote edge scenarios.

The energy savings of 36.5% achieved in our system surpass those in most reviewed studies. This is largely due to our carefully designed reward function, which balances actuation cost and overheating risk. In contrast, many prior methods optimize for thermal constraints alone or lack reinforcement learning altogether.

Moreover, full on-device deployment ensures reliability even in offline environments, as opposed to [29] or [27], which depend on continuous cloud integration or high compute budgets. Our contribution uniquely blends low-latency decision-making with real-time thermal adaptation on a platform with severe computational limitations.

These comparisons confirm that our method sets a new benchmark for lightweight, generalizable, and deployable thermal management in embedded systems.

CONCLUSION

This study presented a deep reinforcement learning-based framework for intelligent thermal management in resource-constrained embedded systems. By modeling the thermal regulation problem as a Markov Decision Process and training a DQN on real telemetry data collected from a Raspberry Pi Zero W, we demonstrated that a learned policy can outperform traditional rule-based fan control mechanisms in both energy efficiency and thermal safety.

The proposed agent successfully learned to anticipate thermal spikes and make proactive decisions regarding fan activation. Unlike hysteresis controllers, which rely on fixed thresholds and lack adaptability, the DQN policy exploited temporal state transitions and reward-based optimization to minimize fan usage while preventing overheating. The inference framework was deployed directly on the target hardware, supported by a custom

GPIO control circuit to actuate the fan, proving its practical viability in real-world environments.

Quantitative results confirmed the model's effectiveness: fan activation time was reduced by over 20% without violating thermal constraints. Confusion matrix analysis showed improved prediction accuracy, and long-run thermal traces validated that the agent maintained CPU temperatures within safe operating bands for over 99% of the test duration. Moreover, the reward progression curves indicated stable convergence, and the system remained robust to thermal load fluctuations during high-CPU activity phases.

Despite the promising results, this work has several limitations that warrant further investigation. The current state representation comprises only two features — CPU temperature and previous fan action — which, while sufficient for the single-device binary control scenario evaluated here, may limit the model's ability to generalize across diverse workload profiles or ambient conditions. Incorporating richer state vectors with features such as CPU frequency, rate of temperature change, or sliding-window temporal embeddings could improve the agent's anticipatory capabilities and robustness to distribution shift. Future work should also evaluate the sensitivity of the learned policy to key hyperparameter choices, including the learning rate, reward shaping coefficients, and exploration decay schedule.

The scalability of this approach to more capable hardware platforms also remains an open question. Extending the system to multi-core processors (e.g., Raspberry Pi 4/5, NVIDIA Jetson) would require expanding the state space to include per-core thermal readings and potentially broadening the action space to support variable fan speeds or per-core settings. Cross-platform transfer learning — training on one device class and fine-tuning on another — presents a promising direction that could reduce the data collection burden for new hardware targets. The lightweight model architecture (two hidden layers of 64 neurons) imposes minimal computational overhead, suggesting feasibility even on significantly more capable platforms.

The combination of anticipatory learning, low-power deployment, and circuit-level integration makes this framework highly suitable for intelligent cooling in IoT devices, edge processors, and embedded robotics. This work paves the way for further exploration into multi-sensor thermal policies, transfer learning across hardware platforms, and the integration of power-aware reinforcement strategies.

ACKNOWLEDGMENTS AND FUNDING SOURCES

The authors received no financial support for the research, writing, and publication of this article.

COMPLIANCE WITH ETHICAL STANDARDS

The authors declare that the research was conducted in the absence of any conflict of interest.

AUTHOR CONTRIBUTIONS

Conceptualization, [O.Y.]; methodology, [O.Y.]; validation, [O.Y., B.K.]; formal analysis, [B.K].; investigation, [O.Y.]; data curation, [O.Y.]; writing – original draft preparation, [O.Y.]; writing – review and editing, [O.Y.]; visualization, [O.Y.] supervision, [B.K.]; project administration, [B.K.].

All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] El Gharbi, M., Abounasr, J., García, R. F., & Gali, I. G. (2024). Textile stretchable antenna-based sensor for breathing monitoring. IEEE Sensors Journal. https://doi.org/10.1109/JSEN.2024.3485472.

[2] Aghaei, M., Fairbrother, A., Gok, A., Ahmad, S., Kazim, S., Lobato, K., Oreski, G., Reinders, A., Schmitz, J., Theelen, M., et al. (2022). Review of degradation and failure phenomena in photovoltaic modules. Renewable and Sustainable Energy Reviews, 159, 112160. https://doi.org/10.1016/j.rser.2022.112160.

[3] Kanellopoulos, D., & Sharma, V. K. (2022). Dynamic load balancing techniques in the IoT: A review. Symmetry, 14(12), 2554. https://doi.org/10.3390/sym14122554.

[4] Berouine, A., Ouladsine, R., Bakhouya, M., & Essaaidi, M. (2022). A predictive control approach for thermal energy management in buildings. Energy Reports, 8, 9127–9141. https://doi.org/10.1016/j.egyr.2022.07.037.

[5] Cao, K., Li, Z., Luo, H., Jiang, Y., Liu, H., Xu, L., Gao, P., & Liu, H. (2024). Comprehensive review and future prospects of multi-level fan control strategies in data centers for joint optimization of thermal management systems. Journal of Building Engineering, 110021. https://doi.org/10.1016/j.jobe.2024.110021

[6] Ahmad, I. (2023). Advances in machine learning for monitoring, control, and optimization of temperature of reactors. https://doi.org/10.20944/preprints202309.1318.v1.

[7] Smith, J. B., & Adams, J. A. (2024). Workload estimation for unknown tasks: A survey of machine learning under distribution shift. arXiv preprint arXiv:2403.13318. https://doi.org/10.48550/arXiv.2403.13318.

[8] Canepa, A. (2023). Application-aware optimization of artificial intelligence for deployment on resource constrained devices. Universita degli studi di Genova. https://doi.org/10.1109/TNSM.2026.3666676.

[9] Garg, N. (2024). Neuromorphic in-memory learning with analog integrated circuits and nanoscale memristive devices [Doctoral dissertation, Universite de Lille; Universite de Sherbrooke, Quebec, Canada]. https://hal.science/tel-04821563/.

[10] Shaheen, K., Hanif, M. A., Hasan, O., & Shafique, M. (2022). Continual learning for real-world autonomous systems: Algorithms, challenges and frameworks. Journal of Intelligent & Robotic Systems, 105(1), 9. https://doi.org/10.48550/arXiv.2105.12374.

[11] Iranfar, A., Terraneo, F., Csordas, G., Zapater, M., Fornaciari, W., & Atienza, D. (2020). Dynamic thermal management with proactive fan speed control through reinforcement learning. In Proc. Design, Automation & Test in Europe (DATE) (pp. 418–423). IEEE. https://api.semanticscholar.org/CorpusID:208524145.

[12] Lin, W., Lin, W., Lin, J., Zhong, H., Wang, J., & He, L. (2024). A multi-agent reinforcement learning-based method for server energy efficiency optimization combining DVFS and dynamic fan control. Sustainable Computing: Informatics and Systems, 42, 100977. https://doi.org/10.1016/j.suscom.2024.100977.

[13] Maity, S., Majumder, A., & Dey, S. (2024). Harnessing machine learning in dynamic thermal management in embedded CPU-GPU platforms. ACM Transactions on Design Automation of Electronic Systems, 27(6), 1–26. https://dl.acm.org/doi/10.1145/3708890.

[14] Kim, S., Bin, K., Ha, S., Lee, K., & Chong, S. (2021). zTT: Learning-based DVFS with zero thermal throttling for mobile devices. In Proc. 19th ACM Int. Conf. on Mobile Systems, Applications, and Services (MobiSys) (pp. 41–53). https://dl.acm.org/doi/10.1145/3458864.3468161.

[15] Zhou, Y., Liang, F., Chin, T.-W., & Marculescu, D. (2022). Play it cool: Dynamic shifting prevents thermal throttling. In Proc. ICML Workshop on Dynamic Neural Networks (DyNN). https://doi.org/10.48550/arXiv.2206.10849.

[16] Tan, T., & Cao, G. (2024). Thermal-aware scheduling for deep learning on mobile devices with NPU. IEEE Transactions on Mobile Computing, 23(12), 10706–10719. https://doi.org/10.1109/TMC.2024.3379501.
[17] Yatskiv, O., & Koman, B. (2025). Assessing the potential of artificial intelligence and machine learning for thermal management in electronic devices. Technology Audit and Production Reserves, 1(81). https://doi.org/10.15587/2706-5448.2025.323117.
[18] Mohammadi, M., & Beitollahi, H. (2022). Q-scheduler: A temperature and energy-aware deep Q-learning technique to schedule tasks in real-time multiprocessor embedded systems. IET Computers & Digital Techniques, 16(4), 125–140. https://doi.org/10.1049/cdt2.12044.
[19] Yeganeh-Khaksar, A., Ansari, M., Safari, S., Yari-Karin, S., & Ejlali, A. (2020). Ring-DVFS: Reliability-aware reinforcement learning-based DVFS for real-time embedded systems. IEEE Transactions on Parallel and Distributed Systems, 31(3), 623–633. https://doi.org/10.1109/LES.2020.3033187.
[20] Akhsham, M., Dousti, M. J., & Safari, S. (2025). Neural network-based control of forced-convection and thermoelectric coolers. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 44(2), 582–591. https://doi.org/10.1109/TCAD.2024.3438689
[21] Tang, J., & Hong, J. (2025). Reinforcement learning-driven task migration for effective temperature management in 3D NoC systems. Scientific Reports, 15, 11933. https://doi.org/10.1038/s41598-025-96335-6.
[22] Kumar, R., & Ghoshal, B. (2022). Machine learning guided thermal management of OpenCL applications on CPU-GPU based embedded platforms. IET Computers & Digital Techniques, 16(6), 308–317. https://doi.org/10.1049/cdt2.12050.
[23] Li, J., et al. (2024). FiDRL: Flexible invocation-based deep reinforcement learning for DVFS scheduling in embedded systems. IEEE Transactions on Computers. https://doi.org/10.1109/TC.2024.3465933.
[24] Liu, H., Yu, J., & Wang, R. (2022). Model predictive control of portable electronic devices under skin temperature constraints. Energy, 260, 125185. https://doi.org/10.1016/j.energy.2022.125185.
[25] Afaq, M., Jebelli, A., & Ahmad, R. (2023). An intelligent thermal management fuzzy logic control system design and analysis using ANSYS Fluent for a mobile robotic platform in extreme weather applications. Journal of Intelligent & Robotic Systems, 107(11). https://doi.org/10.1007/s10846-022-01799-7.
[26] Dzundza, B., Kohut, I., Holota, V., Turovska, L., & Deichakivskyi, M. (2022). Principles of construction of hybrid microsystems for biomedical applications. Physics and Chemistry of Solid State, 23(4). https://doi.org/10.15330/pcss.23.4.776-784.
[27] Ahmadi, A. (2024). EdgeEngine: A thermal-aware optimization framework for edge inference [Doctoral dissertation, University of British Columbia]. https://doi.org/10.1145/3583740.3626616.
[28] Zhang, Z., Zhao, Y., Li, H., Lin, C., & Liu, J. (2024). DVFO: Learning-based DVFS for energy-efficient edge-cloud collaborative inference. arXiv preprint arXiv:2306.01811. https://doi.org/10.48550/arXiv.2306.01811.
[29] Zhang, Q., Zeng, W., Lin, Q., Chng, C.-B., Chui, C.-K., & Lee, P.-S. (2023). Deep reinforcement learning towards real-world dynamic thermal management of data centers. Applied Energy, 333, 120561. https://doi.org/10.1016/j.apenergy.2022.120561.

ПРОГНОЗНЕ КЕРУВАННЯ ТЕПЛОВИМИ ПРОЦЕСАМИ У ВБУДОВАНИХ ЕЛЕКТРОННИХ ПРИСТРОЯХ З ВИКОРИСТАННЯМ ГЛИБОКОГО НАВЧАННЯ З ПІДКРІПЛЕННЯМ

Олег Яцків**, **Богдан Коман
Кафедра системного проектування
Львівський національний університет імені Івана Франка,
вул. Драгоманова 50, 79005, Львів, Україна

АНОТАЦІЯ

Вступ. У роботі представлено підхід на основі глибокого навчання з підкріпленням для інтелектуального керування тепловими режимами у вбудованих електронних системах, орієнтований на енергоефективну та безпечну роботу в умовах динамічних навантажень. Було розроблено спеціалізовану апаратну схему комутації на базі NPN-транзистора для забезпечення керування вентилятором через інтерфейс вводу-виводу загального призначення на платформі з обмеженими ресурсами.

Матеріали та методи. У режимі реального часу сформовано набір даних з використанням одноплатного мікрокомп'ютера Raspberry Pi, який містить показники температури центрального процесора, метрики його завантаження та стани вентилятора протягом 12-годинного контрольованого експерименту. Задачу теплового регулювання змодельовано як марковський процес прийняття рішень, для якого було навчено глибоку Q-мережу (DQN) з метою визначення оптимальної політики активації вентилятора. Навчену модель розгорнуто безпосередньо на пристрої з інтеграцією у спеціалізовану схему керування вентилятором. Виведення виконувалось менш ніж за одну мілісекунду на кожному кроці прийняття рішення з використанням полегшеного середовища виконання PyTorch.

Результати. Результати оцінювання показали, що політика DQN зменшила загальний час активації вентилятора на 23,2% порівняно з базовим правилом гістерезисного керування, забезпечуючи при цьому підтримання температури центрального процесора нижче 60°C протягом понад 99% часу експерименту. Навчений агент активував вентилятор лише у 23,7% випадків, що свідчить про консервативну та енергоощадну стратегію охолодження. Аналіз матриці помилок продемонстрував точність 1,000, повноту 1,000 та F1-міру 1,000 на основі 3442 кроків оцінювання, контрольованих моделлю. Модель правильно ідентифікувала всі 22 події активації вентилятора без помилкових спрацювань та пропусків активації. Порівняльний аналіз із дев'ятьма сучасними підходами на основі штучного інтелекту показав, що запропонований метод забезпечує зниження температури на 11,2°C та економію енергії на рівні 36,5%, працюючи повністю локально без залежності від хмарної інфраструктури.

Висновки. Запропонована модель продемонструвала стійку динаміку навчання, точність прийняття управлінських рішень та проактивну стратегію керування, що мінімізує випадки перегріву. Теплові профілі підтвердили плавність переходів та низьку варіативність параметрів, що свідчить про можливість впровадження політик керування тепловими режимами на основі навчання у реальному часі в периферійних обчислювальних середовищах. Отримані результати формують практичну основу для енергоощадного охолодження та відкривають перспективи створення адаптивного теплового інтелекту у вбудованих системах з обмеженими ресурсами.

Ключові слова: керування тепловими режимами, глибоке навчання з підкріпленням, вбудовані системи, Q-мережа, енергоефективне охолодження, прийняття рішень у реальному часі.

Received / Одержано	Revision / Доопрацьовано	Accepted / Прийнято	Published / Опубліковано
27 February, 2026	18 March, 2026	23 March, 2026	30 March, 2026