Energy-Aware Task Scheduling in Cloud Environments Using Digital Twin Architecture with CNN-LSTM and Multi-Agent Deep Reinforcement Learning

Yong Du
College of General Education, Heilongjiang Polytechnic, Harbin, Hei Longjiang, 150080, China
E-mail: YongDu452@outlook.com

Keywords: cloud energy optimization, digital twin, task scheduling, task allocation, CNN – LSTM, multi-agents DRL

Received: October 27, 2025

The increasing complexity of dispersed cloud networks necessitates smart solutions for energy sustainability and stringent Quality of Service (QoS) maintenance. Existing scheduling approaches, primarily single-agent Deep Reinforcement Learning (DRL), suffer from significant performance and scalability issues in large-scale, dynamic environments. This study proposes a novel, smart, energy-conscious paradigm for task scheduling that integrates a Digital Twin (DT) architecture with Multi-Agent Deep Reinforcement Learning (MADRL). The DT provides a faithful, up-to-the-minute representation of the network, utilizing a hybrid Convolutional Neural Network (CNN)-Long Short-Term Memory (LSTM) model to forecast future workload demands and server power consumption trends. Using these reliable forecasts, MADRL systems develop a dynamic, decentralized scheduling strategy where each server acts as a cooperative agent. The primary objective is to solve a multi-objective optimization problem that minimizes overall energy consumption (via consolidation and dynamic server shutdown) while adhering to QoS requirements (limiting task latency and increasing throughput). Experimental evaluations, comparing our framework against state-of-the-art single-agent DRL baselines (A3C and DDPG), demonstrate superior performance. Our strategy achieves a significant reduction in response time and operational energy cost, proving the viability of this integrated AI and DT technology for efficiently and scalably managing resource allocation in modern distributed cloud environments.

Povzetek: Študija predlaga pametno rešitev za razporejanje nalog v porazdeljenih oblačnih omrežjih, ki z uporabo digitalnega dvojčka in večagentnega globokega učenja zmanjša porabo energije ter hkrati ohranja kakovost storitve (QoS).

1 Introduction

In light of the current explosion in IoT devices and applications, cloud compute offloading services have opened up promising avenues for improved QoS in the next 6G networks [1], [2]. Nevertheless, considering that most IoT devices are often limited in both resources and energy, this difficulty may be mitigated by integrating solutions that are energy efficient [3], [4], [5]. Cloud data centers prioritize energy efficiency because it helps keep costs down and satisfies green computing standards. When energy economy is a factor, the already difficult task of scheduling tasks becomes much more so [6], [7]. Recent studies on energy-efficient task scheduling have focused on two main concerns: scalability and execution overhead. The energy-efficient task scheduling issue has seen extensive usage of machine learning [8], [9], however this approach has mostly focused on resource consumption prediction rather than schedule determination. Nevertheless, we autonomously allotted resources to tasks using the neural network [10], [11]. Our goal in this study was to provide a supervised neural network-based independent task scheduler that might decrease makespan, electrical consumption, execution overhead, and the number of active racks [12], [13]. We achieve our goal by proposing an artificial neural network–based scheduler that, given an incoming job and the existing state of the cloud environment, can forecast which computing resource will be most suitable for completing the task [14] – [18]. We trained our neural network using a massive dataset (about 18 million training examples) generated by a genetic method. We achieved an accuracy rate of 99.9 percent utilizing this dataset in conjunction with the back propagation technique. We tested our energy-efficient task schedulers in both lightly and severely loaded cloud environments using simulations, and we compared them to three popular methods: the genetic algorithm, the MinMIN-MINMin heuristic, and linear regression.

According to the results, the algorithms that were investigated do not perform as well as the suggested approach. Improvements of 64% in makespan, 71% in energy usage, 88% in execution overhead, and 70% in the total number of active racks are achieved in a fully loaded system.

To control the exponential increase of data traffic at the network's periphery in an energy-efficient manner, digital twins are being considered as a potential game-changing technology [19], [20]. We suggest a design for energy-aware task scheduling called a Digital Twin Cloud Network (DTCN). We isolate the task processing time as well as energy use by defining an optimization problem for energy and then finding a set of computing techniques to reduce the former. To accomplish this, we introduce our method for energy-aware task scheduling with the help of digital twins. This approach makes use of both historical and current information at the service and virtualization layers. Next, with the help of physical asset digital twins, IoT devices may either do computation locally or send them to an edge or cloud server. Results from DTCN simulations demonstrate that the suggested energy-aware task scheduling method significantly reduces processing time and energy consumption compared to the state-of-the-art.

1.1 Contributions

When compared to more conventional scheduling approaches, the integrated framework offers substantial improvements:

1. Digital Twin Training for High-Fidelity, Risk-Free Practice: The DT creates an unpredictable and realistic simulation environment where DRL agents are able to acquire non-linear system behaviors (such as cooling-load connections and power of motion draw) without endangering the stability or performance of the real cloud network.

2. Reactive Decision-Making to Proactive Policy Execution using Hybrid Deep Learning: The CNN-LSTM component allows the scheduler to make proactive decisions. The system is able to plan for potential energy spikes or resource bottlenecks by combining the spatial feature collection (correlations across server features) by the CNN with the temporal forecasting (future workload/power needs) by the LSTM.

3. Multi-agent DRL enables decomposition of the scheduling problem's complexity, leading to decentralized global energy optimization (MADRL). The policy for managing energy consumption is more resilient, scalable, and optimal than centralized or heuristic-based approaches because individual agents (servers/clusters) learn to work together toward a global goal—minimizing overall network energy consumption.

4. Optimal Mutual Reduction Achieving a better balance among the two competing goals of decreasing energy expenditures (e.g., by combining virtual machines and shutting down idle hardware) and optimizing quality of service (e.g., by eliminating task latencies and missed deadline) is the main contribution of the framework.

2 Related work

To schedule tasks with energy considerations in mind, the authors of suggested a Digital Twin Edge Network (DTEN) design. We isolate the task processing time as well as energy use by defining an optimization problem for energy and then finding a set of computing techniques to reduce the former. We find a time- and energy-efficient solution using a genetic algorithm-based technique, and we compare it to the Warehouse Location Problem (WLP) since it is NP-hard. To accomplish this, we introduce our method for energy-aware task scheduling with the help of digital twins. This approach makes use of both historical and current information at the service and virtualization layers. Next, with the help of physical asset digital twins, IoT devices may either do computation locally or send them to an edge or cloud server. The suggested energy-aware task scheduling technique is shown to be better in terms of task processing time and used energy in DTEN via the use of simulations [21].

Delays, bandwidth strain, and the potential of a single point of failure are common in traditional, centrally scheduled cloud computing, but inadequate resources are a limitation of single edge computing. Within the context of an edge cloud collaborative architecture, our research zeroes in on technologies that automate scheduling and optimize the distribution of resources. Through the use of”hierarchical awareness-dynamic collaboration" three-tiered edge cloud collaboration architecture, this paper is able to achieve improved resource management and collaboration. The agents at the edge of the network are responsible for resource awareness, while the digital twin engine is located in the cloud. The authors present a hybrid scheduling method called "Dynamic Divide and Reinforcement Scheduling" (DDRS) that uses a feedback reinforcement learning mechanism in conjunction with dynamic divide and conquer of jobs to accomplish scheduling with low latency and high resilience. We build a model of resource allocation that is both cooperative and multi-dimensional; optimize multiple resources at once using the ADMM (Alternative Direction Approach of Multipliers) algorithm and the Nash equilibrium. To handle resource overload, we devise an elastic migration mechanism. Results from experiments demonstrate that

DDRS lowers the average latency to 63 ms with the overtime task rate to 5.3% when compared to pure cloud scheduling and pure edge scheduling combined. Verifying the efficacy of the technology in improving stability, fault tolerance, and efficiency, the resource optimization model decreases the standard deviation of resource utilization from 0.41 to 0.18, decreases the number of elastic migrations to 9, and can recover within 120ms when a node fails. The intelligent upgrading of the full smart factory chain is supported both theoretically and practically by this study [22].

Within the context of an end-edge-cloud collaborative system, this research examines a two-timescale online optimizing for HDT construction. One distinctive aspect of HDT is the idea that VTs for PTs are hosted on edge servers. These VTs may include both generic models that are updated with cloud-based experience knowledge and customized models that are updated with data collected from end devices. Considering the inherent mobility or status variation uncertainties of HDT, we optimize the construction of VTs and PTs' task offloading, as well as the allocation of communication and computation resources, in a joint and dynamical manner to maximize task execution accuracy under strict energy and delay constraints. Our proposed two-timescale accuracy-aware online optimization method, TACO, is based on the observation that decision variables are asynchronous with respect to other triggers. Decomposing the issue into numerous instant ones, TACO uses an upgraded Lyapunov approach. Then, it addresses two timescales alternatively using methods based on block coordinate descent and piecewise McCormick envelopes. The suggested method outperforms its competitors and reaches the asymptotic optimum in polynomial time, according to theoretical studies and simulations [23].

System performance suffers because previous research seldom takes task dependence and service caching into account simultaneously. Due to their shared computation and caching powers, the cooperation of Edge Servers (ESs) should also be considered. This work explores DT-enhanced MEC architecture that takes service cache and task dependence into consideration while intelligently offloading MU duties to collaborative ESs. To that end, we reduce the overall energy consumption of the system by recasting the issue as a MINLP problem. We provide an approach based on Asynchronous Advantage Actor-Critic (A3C) to address this issue. Our suggested technique surpasses the other benchmark methods under many conditions, and extensive simulation results show that it may significantly lower the system's long-term energy usage [24].

When tasks arrive to CEC, a crucial difficulty is task offloading, which determines when and where to execute them. Network outages and underutilization of resources are common outcomes of users' transient connections. Due to a lack of attention in the literature, network congestion and inferior performance have been seen as a consequence of simultaneous mobility-aware dependent task outsourcing with network flow scheduling. In response, we develop a combined online mobility-aware dependent task offloading as well as bandwidth allocation issue to enhance service quality via reduced energy consumption and task completion times. Our new method, MDT-DRL, is a Digital Twin-assisted Deep Reinforcement Learning system that takes mobility into consideration. In order to adapt to the mobile CEC system, our digital twin model provides future user states to the reinforcement learning process, which in turn allows for effective offloading plans. In terms of average task completion as well as energy consumption, experimental findings on both synthetic and real-world datasets demonstrate that MDT-DRL outperforms state-of-the-art baselines [25].

With the goal of effectively and reliably performing intelligent driving tasks, we build the architecture of Digital Twin empowered Cloud-native Vehicular Networks (DT-CVN) and the process for task execution in this study. Within DT-CVN, we offer a framework for digital twin design and implementation that allows for the distributed deployment of various modules within an identical digital twin entity. This is achieved by leveraging the distributed attributes of cloud-native microservices. We also develop methods for digital twins invocation for job scheduling in DT-CVN, which may further increase its efficiency, and for reusing modules. Next, we provide an adaptive task scheduling technique based on deep reinforcement learning (DRL) by modeling the task scheduling in DT-CVN as a combinational optimization problem. According to the simulation findings, the suggested approach may decrease energy usage while improving work scheduling efficiency [26].

In this study, we introduce a new distributed architecture that may enhance energy efficiency in industrial settings that use a lot of energy by creating digital twins of existing systems. In order to optimize industrial processes, our platform can execute user-defined workflows that include real-time monitoring, forecasting, with simulation microservices. This will improve decision-making techniques. An Apache Kafka-based backbone powers the platform's stateless centralized orchestration engine, which guarantees scalability, fault tolerance, and effective handling of heterogeneous data. Quickly and easily configure workflows, communicate asynchronously to speed workflow execution, and manage tasks dynamically and based on events using API-driven scheduling. To demonstrate its efficacy across many energy management issues, this platform will be implemented and tested across multiple energy-intensive industrial settings, assisting with the control of various facilities' energy systems [27].

To achieve the strict quality of service (QoS) criteria, it has been essential to offload computer-intensive processes to the mobile edge servers, because these devices have limited resources. Achieving effective edge computing services is complicated, however, since there are many different kinds of smart cars, each with its own set of features, needs, and topology. In order to address these issues, we implement a connected vehicular network that is enabled by digital twins (DT). This network can improve the quality of service for vehicle users by automatically managing their resources. In particular, we want to devise the best offloading mechanism and distribute computing as well as spectrum resources for a vehicular network that is supported by DT. To further increase throughput, decrease total delay, and optimize energy usage, a task is defined to optimize the utility-cost ratio (UCR) of the vehicular network. Additionally, we provide a three-stage iterative approach to get an ideal framework for allocating resources by optimizing bandwidth, latency pricing, energy usage, and the offloading factor until convergence. In addition to enhancing throughput, the simulations show that the proposed method effectively cuts average energy use for vehicular networks, speeds up task offloading, and speeds up overall network performance. At a vehicle population density of 0.137 vehicle/m, the utility-to-cost ratio drops by 85% in a sparse traffic scenario and 12.5% in a congested traffic scenario [28].

Through coordinated resource management, we examine the global loss function reduction issue under the long-term AoI constraint in this study. The AoI-DT intelligent resource management method is suggested as a solution to the optimization issue, which is decoupled using telescoping sum with Lyapunov optimization. Through the use of device scheduling with channel allocation, AoI-DT is able to strike a compromise between the AoI guarantee and the accuracy of the EDC model. When compared to two state-of-the-art algorithms, the simulation results show that AoI-DT performs better in terms of the global loss function and AoI [29].

The potential for very effective resource scheduling in DTs is presented by edge computing, a distributed computing architecture. This gap is the driving force for this paper's attempt to address the challenge of updating and modeling high-fidelity DTs in real-time. We begin by creating a Heterogeneous Computing Task Graph (HCTG) to depict the DTs' computing activities. Afterwards, a Hierarchical Attention Technique (HAT) is suggested for retrieving the HCTG's latent representation fields. In order to meet the minimal total completion time requirements of various DTs, we provide a Deep Reinforcement Learning (DRL)-inspired computing task scheduling technique, HAT-DRL, which is based on our Markov Decision Process (MDP). The suggested technique outperforms competing task scheduling algorithms, and experimental findings show that it has promising scheduling efficiency [30]. Table 1 shows the comparison of the digital twin enabled energy aware task scheduling and resource optimization technique.

Table 1: Comparison of digital twin–enabled energy-aware task scheduling and resource optimization methods

Study	**Architecture / System Model**	**Optimization / Scheduling Method**	**Role of Digital Twin (DT)**	**Objectives**	**Key Results / Performance**
Energy-Aware Task Scheduling in DTEN [21]	Digital Twin Edge Network (DTEN); service + virtualization layers; IoT offloading to edge/cloud	Genetic Algorithm (GA); compared with Warehouse Location Problem (WLP)	DT models physical assets; uses historical & real-time data for energy-aware scheduling	Minimize energy + task processing time	GA-based DT scheduling reduces energy use and task delay over conventional methods
Hierarchical Edge–Cloud Collaboration with DDRS [22]	3-tier edge–cloud architecture with hierarchical awareness	Dynamic Divide and Reinforcement Scheduling (DDRS) + ADMM multi-resource optimization	Cloud-level DT engine maintains global state for edge agents	Low latency, fault tolerance, resource balance	Latency reduced to **63 ms**, overtime tasks to **5.3%**; resource utilization std dev drops **0.41 → 0.18**
Two-Timescale HDT Optimization (TACO) [23]	End–edge–cloud system with Virtual Twins	Two-timescale accuracy-aware online optimization	DT (VT+PT) captures mobility, dynamic states,	Maximize accuracy under	Achieves asymptotic optimality; outperforms

	(VTs) hosted on edge servers	using Lyapunov + BCD	and energy/delay constraints	delay/energy constraints	baselines in energy-delay-accuracy tradeoff
DT-Enhanced MEC with Service Caching + Task Dependence [24]	Mobile Edge Computing (MEC) with cooperative Edge Servers (ESs)	MINLP solved with A3C (Asynchronous Advantage Actor–Critic)	DT models ES cooperation, cache states, task dependencies	Minimize long-term system energy	A3C + DT significantly lowers long-term energy vs benchmarks
Mobility-Aware Dependent Offloading with MDT-DRL [25]	Client–Edge–Cloud (CEC) with user mobility	Digital Twin–assisted Deep Reinforcement Learning	DT predicts future user mobility + network states	Reduce energy + task completion time in mobile systems	MDT-DRL outperforms state-of-the-art in delay + energy consumption
DT-CVN for Vehicular Networks [26]	DT-enabled Cloud-Native Vehicular Networks (DT-CVN)	DRL-based adaptive scheduling	DT entity distributed as microservices; invoked for job scheduling	Reduce energy; improve task execution for vehicles	Improves throughput; reduces energy-use and delay in vehicular tasks
Distributed DT Platform for Industrial Energy Systems [27]	DTs for industrial systems with Kafka-based orchestration	Event-driven dynamic workflow scheduling	DT predicts, simulates, and optimizes energy workflows	Improve industrial energy efficiency	Validated across multiple industrial environments; enhances energy decision-making
DT-Enabled Vehicular Offloading Optimization [28]	DT-assisted connected vehicular network	3-stage iterative optimization	DT provides resource state and workload predictions	Maximize utility-to-cost ratio (UCR)	UCR improved; energy and delay reduced; 85% improvement in sparse scenarios
AoI-DT Intelligent Resource Management [29]	Global model with Age of Information (AoI) constraints	Lyapunov-based optimization + device scheduling	DT keeps global AoI state and model accuracy	Reduce global loss + maintain AoI guarantee	Outperforms baselines in AoI + global loss reduction
HAT-DRL for DT Scheduling [30]	DT with Heterogeneous Computing Task Graph (HCTG)	Hierarchical Attention Technique (HAT) + DRL	DT maintains HCTG representation and high-fidelity updates	Minimize total completion time	HAT-DRL achieves superior scheduling efficiency vs SOTA

3 Architectural framework

Problem Definition: Achieving a balance between energy consumption, job completion delay, and safety risk is achieved by minimizing the weighted total unit cost.

$$\min \sum_{t=1}^{T} (\alpha \cdot L_t + \beta \cdot E_t + \gamma \cdot R_t)$$

Where:

- T constitutes the whole-time horizon, for instance, the quantity of scheduling intervals.
- L_t : The total amount of time it takes for all tasks to be executed t.
- E_i : Total power used by all orbiting satellites for data processing and transmission at any one moment t.
- R: Decisions on the distribution of tasks at the moment are correlated with a Fuzzy Safety Risk Score t.
- α, β, γ : The ranking of Latency, Energy, or Risk may be adjusted using user-defined weighting factors, in that order ($\alpha + \beta + \gamma = 1$).

A high-fidelity feedback loop connecting the real and virtual cloud infrastructures forms the basis of the system, which is given in fig 1,

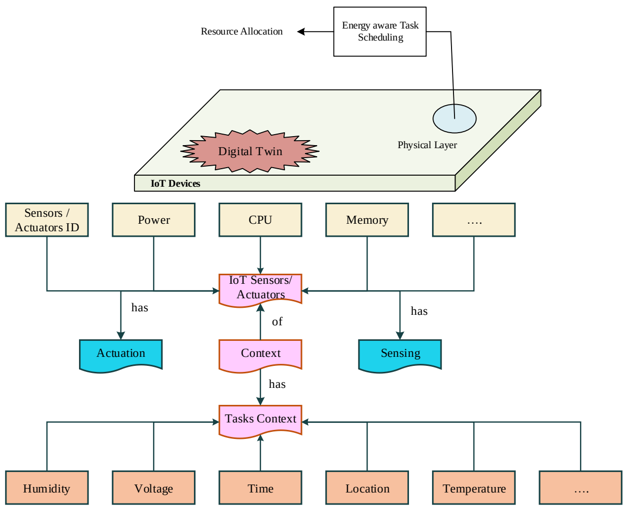

Figure 1: Digital twin model for cloud network energy optimization

1. The Physical Layer (The Real Cloud)

What you see here is the real deal when it comes to cloud data centers.

- Hardware: Servers (central processing units, graphics processing units), storage, network switches, and PDUs are all examples of hardware.
- Sensors: These keep an eye on things like temperature, power consumption (Watts), CPU/GPU use, and network traffic in real time.
- Orchestrator: Cloud management software like Kubernetes or OpenStack acts as the orchestrator, physically carrying out the duties.

2. The Digital Twin Layer (The Standardized Sandbox)

The Physical Layer is being simulated in real-time.

- A virtual environment simulates the real-life conditions of all hardware components, including servers, network links, and cooling units.
- You may do quick "what-if" analyses with the help of the simulation engine. It can execute millisecond-long simulations of scheduling decisions' energy and performance impacts (such as "What happens if I run Task A on GPU-Server 10?") without affecting the actual hardware.
- The "Standard" one: We will use this DT as our standard. To provide a fair comparison, this standardized and repeatable virtual environment is used to test and benchmark any new optimization strategy (from the DRL).

3. The Predictive Layer (CNN + LSTM)

It is the responsibility of this layer inside the Digital Twin to comprehend the present condition of the cloud and to foretell its future condition.

- A CNN examines the data center's physical location.
- Data center temporal (time-series) state analysis is performed using LSTM (Long Short-Term Memory).

4. The Optimization Layer (Multi-agent DRL)

It is this "brain" that decides when things should happen. It learns the most effective energy-saving policies using CNN/LSTM predictions and the Digital Twin's safe sandbox.

- Agents: a group of DRL agents that work together or against each other.
- Policy: every agent's "brain" that converts a "state" into an "action."
- Reward Function: The aim that is standardized: the reward function. When agents accomplish their goals while using the least amount of energy possible, they will get rewards.

3.1 Task model

In order to predict how long it will take for various RNN jobs to complete, a performance model must be constructed. Predicting how long each job will take to complete using the task model allows for more informed decisions about how to distribute work. We started by gathering information from the training sequences in order to construct a task model. To measure the variances, we used n various lengths of input sequences as well as repeated each length m times. Keep in mind that several CPU running frequencies occurred because of DVFS. Imagine a world where p frequency modes exist. This means that we have n×m×p sets of input patterns, while the j th time for the k th frequency, the input sequences are repeated π th length $d_{l,j,k}$, which yields the running time $t_{l,j,k}$. Therefore, we attained data tuples $< d_{ij,k}, t_{l,j,k} >, 1 \leq i \leq n, 1 \leq j \leq m, 1 \leq k \leq p$ in the data collection. In order to get the task model, we trained it using the acquired data. We constructed a linear performance framework in the following way since the time it takes to complete an RNN inference job is almost directly proportional to the length of the input:

$$T_k(d) = a_k \cdot d + b_k + c_k \qquad (1)$$

where T_k is the expected duration of the RNN task's execution, where d is the input sequence length, a_k is the bias coefficient, c_k is the additional time to switch frequencies, and subscript T_k indicates the kth operating frequency; this is done in accordance with the procedures outlined. We used data gathering and the linear least squares regression approach to get the coefficients of Equation (1). $\langle d_{l,j,k}, t_{l,j,k} \rangle$. Then, we found

$$a_k = \frac{m \cdot n \cdot \Sigma_{k,j}(d_{t,j,k} \cdot t_{t,jk}) - \Sigma_{k,j} d_{t,jk} \cdot \Sigma_{k,j} t_{t,jk}}{m \cdot n \cdot \Sigma_{k,j} d_{t,jk} - (\Sigma_{k,j} d_{t,k})^2} \qquad (2)$$

$$b_k = \frac{\Sigma_{t,j} t_{t,jk} - a_k \cdot \Sigma_{t,j} d_{t,jk}}{m \cdot n} \qquad (3)$$

Equations (2) and (3) were used to derive the k-th operational frequency regression model that was utilized for work allocation. Here is how we got a task design T with all the frequencies:

$$\mathbb{T} = \{T_k \mid 1 \leq k \leq p\} \qquad (4)$$

3.2 Task Allocation

We learned that frequency impacts task energy from our findings in Section 3. At the k-th operational frequency, we constructed an energy model to lower the power consumption of cloud computing systems running at the edge:

$$E_k = T_k \cdot P_k \qquad (5)$$

where e_k is the task energy, T_k is the power at the kth operating frequency and is equal to the running time of the inference job. Afterwards, we calculated the energy model for every frequency in the following way:

$$\mathbf{E} = \{E_k \mid 1 \leq k \leq p\} \qquad (6)$$

After that, we came up with a method for dividing up the work as follows:

$$S(d,q) = \begin{cases} \text{Edge} & \text{if } \exists T_k \in T, T_k(d) \leq q \\ \text{Cloud} & \text{otherwise} \end{cases} \qquad (7)$$

Thus, S is the function that allocates tasks, d is the sequence of input data, while q is the quality-of-service criterion, which is the maximum acceptable task delay. Once the task allocation mechanism determines that the edge devices can meet the Q0S criteria, the task is executed there. If not, it will forward the work to the server in the cloud. We used DVFS methods to dynamically modify the processor frequency while doing activities on edge devices in order to decrease energy usage. Here's how it worked:

$$\underset{E_k}{\operatorname{argmin}} f(E_k) = \{E_k \mid E_k \in \mathbb{E}, T_k \leq q\} \qquad (8)$$

To reduce energy, we chose the lowest working frequency that nevertheless matched the quality-of-service criteria. The computation of the overhead time in Equation (1) was done in the following manner when we changed the CPU speeds from the π operating rate to the k th operating frequency:

$$c_k = \begin{cases} 0 & \text{if } i = k \\ C & \text{otherwise} \end{cases} \qquad (9)$$

wherein C is a constant, following the procedure outlined in Section 6.2.2. Based on the most recent data, we revised the energy model after including frequency variation with DVFS approaches. We used live data to update the coefficients in Equation (1) so that the model constantly represented the current edge device characteristics. An advanced, standardized system for improving cloud computing's energy-aware job scheduling is laid forth in this proposal. For its central simulation and standardization environment, it employs a Digital Twin (DT) architecture. The DT receives its predictive insights from convolutional neural networks (CNNs) and long short-term memories (LSTMs), and its optimization engine for scheduling choices is Multi-agent Deep Reinforcement Learning (MARL).

Objective Function: The main objective is to determine the best scheduling strategy, denoted as σ ", that reduces, over a given time horizon T, the weighted total of energy consumption (E) and performance penalties (P), such as service level agreement (SLA) breaches.

$$\pi^* = \arg\min_{\pi} \mathbb{Z}\left[\sum_{t=0}^{T} \left(w_e \cdot E_t(\pi) + w_p \cdot P_t(\pi)\right)\right]$$
(10)

Where:

- w_c and w_p strike a balance between efficiency and effectiveness in terms of weights.
- E_t is the sum of all energy used at the current instant.
- P_t is the cost that is incurred at time t due to service level agreement breaches, such as a latency greater than 100 ms etc.

3.3 Convolutional layers for spatial feature extraction

Convolutional layers handle the input feature tensor in the initial stage of the hybrid architecture $X \in \mathbb{R}^{n \times d \times 1}$, the channel is represented by the last dimension, d is the total number of features, and n is the number of samples. In order to extract spatial patterns, these layers use learnable filters W. The calculation for the convolution operation is (4):

$$Z[i,j] = \sum_{k=1}^{k_w} X[i,j+k] \cdot W[k] + b, \qquad (11)$$

where k_w where b stands for the bias term, W for the filter weights, while is the size of the kernel. To capture feature dependencies at the local level, this procedure slides the filter through the input tensor. A non-linear activation function is applied to the convolutional output. To make understanding complicated connections even more flexible, this design swaps out the usual ReLU with a bespoke activation function, as seen in (12):

$$\mathrm{CustomActivation}\ (z) = \tanh\ (z) \cdot \sigma(z) \qquad (12)$$

Wherein the custom activation equation is given by (6) and (7) and is a mix of the tanh or sigmoid activation functions:

$$\tanh\ (z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \qquad \sigma(z) = \frac{1}{1+e^{-z}} \qquad (13)$$

The characteristics of tanh, which permits outputs within the interval (-1,1), and σ, which condenses data into (0,1), are combined in this customized activation function. A non-linear, smooth function is produced, which improves the model's capacity to learn about small changes in the input. To decrease the spatial dimensions of the feature maps, max-pooling is used after convolution and activation, as shown in equation (8):

$$Z_{\mathrm{pooled}}\ [i,j] = \max_{k\in[J/t+p]} Z[i,k] \qquad (14)$$

with the pooling size denoted by p. Reducing computational complexity without sacrificing crucial information, this stage concentrates on the most significant aspects.

3.4 LSTM layers for temporal dependency modeling

An LSTM layer models temporal relationships by flattening the feature maps supplied from the CNN layers. The LSTM cell uses gating mechanisms to update two states—one visible to the cell at time step t and one hidden state $\mathbf{h}_t$. As specified in (9) through (11), forget gate, the input gate, as well as output gate are thus:

$$\mathbf{i}_t = \sigma(\mathbf{W}_t[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_t) \qquad (15)$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \qquad (16)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \qquad (17)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activation function, and W_t, W_f, W_o are weight matrices that can be trained. The concealed state and cell state have been revised as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(\,W_c[\,h_{t-1}, x_t] + b_c) \qquad (18)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \mathrm{CustomActivation}\ (c_t) \qquad (19)$$

In this case, the network's capacity to represent intricate temporal patterns is improved by modifying the cell state output using the custom activation function, which is given in Fig 2.

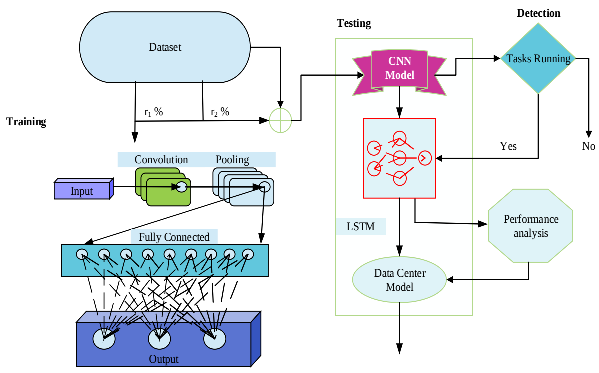

Figure 2: CNN-LSTM model for spatial feature extraction and temporal feature extraction

Dense output layer and prediction

The output of the LSTM layer, h_{LSTM}, for binary categorization by use of a thick layer. Final forecast as computed by the dense layer is (14):

$$\hat{y} = \mathrm{CustomActivation}\ (W_o \cdot\ h_{LSTM} + b_o) \qquad (20)$$

where W_o and b_o represent the biases and weights of the thick layer. With the help of the unique activation function in the output layer, the algorithm is able to provide probabilities that are well-calibrated and sensitive to small changes in the input data.

Custom loss function

During training, a loss function that incorporates both mean squared error and binary cross-entropy (BCE) is minimized. This is the definition of the loss function (15):

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{t=1}^{n}\left[-y_t \log\ (\hat{y}_t) - (1-y_t)\log\ (1-\hat{y}_t) + \frac{\lambda}{2}(\hat{y}_t - y_t)^2\right] \qquad (21)$$

The wrong predictions are penalized by the binary cross-entropy term, as shown in equation (16):

$$\mathrm{BCE}(\hat{y}_x, y_t) = -[y_t \log\ (\hat{y}_t) + (1-y_t)\log\ (1-\hat{y}_t)] \qquad (22)$$

Following that, as shown in (17), the mean squared error term pushes the projected probability closer to the actual labels:

$$\mathrm{MSE}(\hat{y}_t, y_t) = \frac{1}{2}(\hat{y}_t - y_t)^2 \qquad (23)$$

The two parts are balanced by the regularization parameter Λ, which guarantees both precise classification and probabilistic calibration. During model training, the Adam optimizer takes the second moments and gradients into account to fine-tune the learning rate of each parameter. The rule for updates is provided by (24):

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\nabla\mathcal{L}(\theta_t)}{\sqrt{v_t}+\epsilon'} \qquad (24)$$

where η is the learning rate, $\nabla\mathcal{L}(\theta_t)$ denotes the loss function's gradient, v_t stands for the average of squared gradients with exponential decay, and ϵ is a tiny constant that ensures numerical stability. The hybrid CNN-LSTM design has a dense output layer for prediction, LSTM layers for modeling temporal dependencies, and convolutional layers for extracting spatial features. By including a personalized activation mechanism, the model's capacity to acquire intricate

3.5 MADRL

Agents in MARL are the "schedulers" that figure out the best course of action.

- **Agents:** This is a hierarchical system:
  - The master global scheduler agent can see all cloud states, including those from CNNs and LSTMs. "A fresh training job arrives; it will go to Rack 10." is an example of a high-level choice it makes.
  - One example of a node agent is a worker. Every server or server rack has its own "agent." After receiving the job from the Global Agent, it takes little judgments such as "I will run this process on GPU 3" or "I am idle; I will go into a low-power sleep state."
- **State:** The agents' "state" is the sum of the CNN's (spatial info) with the LSTM's (temporal info) outputs.
- **Action:**
  - **Task Placement:** (task_T, server_S, GPU_G)
  - **Power Management:** (server_S, power_state_P) (e.g., C-states, P-states)
  - **Data Routing:** (data_D, network_path_N)
- A learning algorithm (such as Multi-Agent Q-Learning) may estimate the future total reward using a Qfunction Q_i (s,a) that is learned by every agent i (or a central critic). The agents use the Bellman equation to update their Q-values as they investigate activities in the DT.

$$Q_i(s,a) \leftarrow Q_i(s,a) + \alpha\left[R_{t+1} + \gamma\max_{a'} Q_i(s_{t+1},a') - Q_i(s,a)\right] \qquad (25)$$

This is all taking place inside the standardized Digital Twin, so the policy that comes out of it is both very optimized and shown to be energy efficient when measured against a common, repeatable standard. Reward: Cutting down on energy usage and average processing time for tasks is the holy grail of optimization problems. The time and energy needed to analyze tasks and execute them are therefore factors that reward synthetic considers. $r(s_t,a_t) = (\alpha T_t)^{-1} + (\beta E_t)^{-1}$ is used to symbolize the payout during time period t, where $T_t = \sum_k T_{t,k}^{proc}$ is the sum of time delay and $E_t = \sum_k E_{t,k}^{proc}$ include all energy expenditures incurred while processing jobs within time slot t. So, when we consider state s with policy π, the value function of V is:

$$V^{\pi}(s) = \mathbb{E}[\Sigma_0^{\infty} r(s_t,a_t) \mid s_0 = s] \qquad (26)$$

The discounted factor γ determines the ideal scheduling strategy π^*.

$$\pi^*(s) = \arg\max_a \sum_{s'} p(s' \mid s,a)\left[r(s,a) + \gamma V^{\pi^*}(s')\right] \qquad (27)$$

The system operates in a continuous loop in this way:

1. Observe: The Digital Twin receives data from the Physical Layer's real-time sensors, including power, temperature, and load.
2. Update: The Digital Twin revises its internal state until it is an exact replica of the live cloud.
3. Predict: By analyzing present geographical hotspots, the CNN determines future workload, while the LSTM forecasts it.
4. Pay attention: The MARL agents get this state that combines the present state with the expected future state.
5. Run the numbers: The MARL agents use the simulation engine of the Digital Twin to run thousands of different scheduling scenarios before making a final decision. With each command, the DT immediately returns the "consequence" (energy, latency).
6. Acquire knowledge: The agents revise their policies based on the activity that yields the largest simulated reward.

4 Performance evaluation

In this part, we compare the proposed method against a number of benchmark algorithms and use numerical tests to determine how well it performs.

4.1. Experiment environment and setting

The system requirements are for an Intel i7-13700k CPU and an NVIDIA RTX4090-24G GPU, with Python 3.7 and TensorFlowGPU-1.14.0 rounding out the software stack. The following is the configuration of the DNN parameters. The dimensionality of potential actions determines the number of neurons for the third hidden layer, whereas the first two layers are configured with 300 and 100 neurons, respectively, for Actor. All three of Critic's hidden layers have 300, 100, and 1 neurons, respectively. The beginning value as well as reducing rate for investigation are set to ϵ0 = 0.9 as well as β = 10−4, respectively, while the learning rates of the Actor as well as Critic networks are set to γa = 10−4 or γc = 10−3, with the discount factor set to γm=0.9, which is given in Table 1.

Table 1: Simulation parameters and descriptions

Category	**Parameter**	**Value / Range**
Digital Twin Parameters	Synchronization Frequency	Every 10 s
	Energy Model Accuracy	±5% deviation
	Thermal Model Type	RC-network / Data-driven
Predictive Model (CNN–LSTM)	Input Window Size	10 – 30 steps
	CNN Filter Size	3 × 3
	CNN Layers	2
	LSTM Hidden Units	256
	Batch Size	64
	Learning Rate	1e-3
	Loss Function	Weighted MSE + MAE
Multi-Agent DRL Parameters	Number of Agents	50 – 100
	Algorithm	MAPPO / MADDPG
	Discount Factor	0.96
	Actor Learning Rate	3e-4
	Critic Learning Rate	1e-4
	Replay Buffer Size	1e8
	Reward Weights	(1.0, 10.0, 0.5, 2.0)
	Exploration Strategy	ε-greedy / entropy regularization
Evaluation Metrics	Total Energy Consumption	kWh
	SLA Violation Rate	%
	Task Completion Time	s
	Migration Cost	ms or %
	Thermal Violation Count	—
	Resource Utilization	%
Environment Configuration	Number of Physical Hosts	50 – 500
	Number of Virtual Machines (VMs) / Tasks	200 – 2000
	Host CPU Capacity	2.0 – 3.7 GHz
	Memory Capacity	8 – 133 GB
	Network Bandwidth	1 – 10 Gbps
	Simulation Duration	24 h (1 day)
	Time Step Interval	1 – 4 min
	Data Sampling Rate	1 record/min
Workload & Power Model	Workload Type	Mixed (CPU, I/O, Memory)
	Workload Arrival Distribution	Poisson
	CPU Utilization Range	10% – 100%
	Idle Power Consumption	50 – 70 W
	Peak Power Consumption	190 – 270 W
	Power Model Equation	k=1.5–2.5k = 1.5 – 2.5k=1.5–2.5
	Temperature Threshold	80 °C

4.2. PMV dataset collection and tasks generation

We pre-collected the information concerning occupant thermal comfort from multi-apartment smart houses in Jeju city, South Korea, for eleven months (330 days), and we store it in the supervisor server's database. A system of sensors with smart meters installed at various locations around the houses allowed this to happen. While data about the outside environment is retrieved from online services, these sensors successfully recorded critical factors including energy use, air quality, humidity, and temperature within the intended smart home setting. Data collection jobs are then produced and performed at edge gateways. Instead of gathering data from IoT sensors, the tasks query a database on the server to get the information. Data is collected at different time periods by means of the produced jobs. For data collection over 330 days, for instance, 47520 tasks may be created utilizing 10-minute intervals on a daily basis. Info about indoor temperature (IT), optimized indoor humidity (OIH), indoor humidity (IH), optimal indoor temperature (OIT), outside temperature (OT), outdoor humidity (OH), and heater power (HP) is required for the inquiry data.

4.3. Performance metrics

Queuing Delay: A microservice will create a queue if the number of incoming tasks is greater than its maximum handling capacity. One cannot disregard the delay if the job is awaiting execution in the queue. Queuing theory states that in $ms_{s,m}, Q_{t,s,m}$ stands for the length of the queue at time t, meaning the total number of jobs waiting to be executed $ms_{s,m}$. The queue length at time $t + 1$ is:

$$Q_{t+1,s,m} = \max\{Q_{t,s,m} - \phi_{t,s,m}, 0\} + \Delta Q_{t,s,m} \tag{28}$$

where $\phi_{t,s,m}$ represent the activities that have been handled inside the last time slot t and $\Delta Q_{t,s,m}$ the assignments that have just come in. Assuming that λ is the average queue service rate, for the given job

$$T_{t,k}^{\infty} = \frac{Q_{t,s,m}}{\lambda} \tag{29}$$

Using the binary variable, we can show that the job is running or not $y_{k,s,m}$ to show whether the task $V_{t,k}$ is in the queue ($y_{k,s,m} = 0$) or is being deal with ($y_{k,s,m} = 1$).

Energy cost model: Calculating the computational energy use when performing tasks locally as well as the transmission energy consumption while transferring the job for a vehicle are both important to lower the energy consumption on the cars. Performing the $V_{t,k}$ computing power used by vehicle k while operating locally is:

$$E_{t,k}^{rex} = P_k^{rxe}\left(f_k^{le}\right) \cdot T_{t,k,le}^{rxe} = \frac{P_k^{rxe}\left(f_k^{le}\right) \cdot D_{t,k} C_{t,k}}{f_{t,k}^{le}} \tag{30}$$

When the task $V_{t,k}$ transported to RSU for processing through vehicle k, the amount of energy used during transfer will be:

$$E_{t,k}^{tr} = P_{t,k}^{tr} \cdot T_{t,k,r}^{tr} = \frac{p_{t,k}^{tr} D_{t,k}}{R\left(p_{t,k}^{tr}\right)} \tag{31}$$

Hence, the amount of energy needed by the vehicle k to complete the task $V_{t,k}$ could be:

$$E_{t,k}^{tot} = \begin{cases} E_{t,k}^{ex}, & \text{if } d_{t,k} = 0 \\ E_{t,k}^{tr}, & \text{otherwise} \end{cases} \tag{32}$$

We build data-gathering and PMV-optimization jobs of varied sizes so that we may compare their scheduling performance. Data is being requested from the server's database and processed for additional optimization during a time period (in minutes) that is represented by the size of a task. As a workload, many jobs are generated with time intervals ranging from 1 to 5,000. The amount of time it takes to execute $T_{i,j}$ for task T_i by use of a digital item (edge gateway) When we divide the whole size of the work through the processing speed of V, we get V_j. Hence, the total execution time may be determined in the following way, with $c_{i,j}$ serving as the decision variable for the task's whether or not the T_i is allocated to V_F

$$\text{TotalEvecutionTime} = \sum_{i \in V} \sum_{j \in J} c_{i,j} T_{i,j} \tag{33}$$

Fig displays the outcome of calculating the overall execution time required to plan the specified number of randomly chosen task's $V_{t,k}$, the queuing time $T_{t,k}^{w}$:

4.4 Comparing algorithms

We choose two popular heuristic scheduling methods that have been tested and found to be both simple and successful, taking into consideration both our scenario's unique requirements and the established norms of relevant literature. When compared to heuristic algorithms, the suggested approach clearly outperforms them in terms of efficiency and performance.

- One popular scheduling strategy for scheduling tasks over many processors is Short Job First (SJF). When it comes time to schedule the application, the SJF algorithm prioritizes jobs based on the amount of their data.
- FCFS, or First Come, First Serve, is another popular scheduling method. It guarantees fairness in task scheduling by processing jobs in arrival order. Based on the current status of the system,

both FCFS as well as SJF randomly allocate available microservices to jobs.

4.5 Discussion

To reduce data center energy usage without compromising service-level agreements (SLAs), the authors of the suggested framework "Energy-Aware Task Scheduling with Optimization in Cloud Computing using Digital Twin Architecture with CNN, LSTM, with Multi-Agent Deep Reinforcement Learning (DRL)" set out to find a solution. With the help of a Digital Twin (DT), cloud infrastructures may be accurately simulated in real-time, creating a risk-free setting for training and testing models. To ensure system stability, energy economy, and balanced task distribution in large-scale cloud settings, this system uses intelligent agents and deep learning. The Digital Twin is the central component of this design; it is a scale model of the cloud that mimics its physical and operational features, such as its servers, networking, cooling systems, as well as energy consumption patterns. Prior to deploying to the actual infrastructure, the DT is used as a testbed to simulate job scheduling choices. As a result, there is less room for error and more room for continuous improvement without interrupting service. Ensuring realism and responsiveness is achieved by the ongoing synchronization of real-time telemetry data among the physical as well as virtual environments. This data includes things like CPU usage, temperature, task type, and power consumption. In this setup, predictive analytics are handled by the CNN-LSTM hybrid model. In order to pinpoint areas with excessive heat or power consumption, the CNN maps out the spatial connections between physical nodes. At the same time, LSTM networks learn from past workload variations, power consumption patterns, and thermal dynamics to interpret temporal dependencies. When used in conjunction, these models may predict fluctuations in energy usage and workload demand over the next several days, allowing for more deliberate deployment of resources. In order to optimize the migration and scaling of tasks and to anticipate resource bottlenecks, the Digital Twin as well as Multi-Agent system is guided by the predictive insights from the CNN-LSTM network. This framework's decision-making mechanism is the Multi-Agent DRL system. Every agent is trained to schedule tasks and manage power efficiently by being allocated to a particular host, rack, or resource cluster. Agents work together in an integrated setting to accomplish optimization goals on a global scale by monitoring their local states, which include things like CPU load, energy consumption, and expected demand. Agents use state-of-the-art algorithms like MAPPO or MADDPG to combine centralized training with decentralized execution when it comes to optimizing policies. This guarantees minimal computing overhead, scalability, and flexibility to changing workload circumstances. Energy reduction, job completion duration, and SLA adherence are all carefully considered in the design of the DRL framework's reward mechanism. Maintaining performance effectiveness within thermal as well as energy thresholds is rewarded, whereas excessive energy usage, SLA breaches, and superfluous task migrations are penalized. Based on demand forecasts and temperature conditions, the system learns to optimize overall power use by altering CPU frequencies (DVFS), scaling virtual machines, and dynamically consolidating workloads.

To start, agents learn rules in a simulated DT environment, which doesn't impact the actual cloud infrastructure in any way. To improve their situational awareness, the DRL agents are regularly fed short-term predictions by the CNN-LSTM model. You may deploy the models to the live system after they reach stable performance. The DT will then offer a parallel validation environment so you can fine-tune and verify safety in real-time. The physical structure, digital twin, and educational agents work together in a closed-loop to build an ecosystem that can change its schedule on its own.

Benchmarking against more conventional schedulers like round-robin, first-fit reducing, and heuristic energy-aware methods would be an important part of any experimental assessment of this design. Total usage of energy, average rate of service level agreement violations, time to completion of workloads, and number of migrations are all key performance metrics. Reduced energy consumption and thermal violations, enhanced load balancing, and robust system performance are the anticipated results. To sum up, this standardized framework offers a strong, smart, and flexible method for optimizing and scheduling tasks in the cloud while taking energy into consideration. It offers real-time optimization that adapts to operational conditions while assuring sustainability and performance by integrating predictive deep learning (CNN-LSTM), intelligent coordination (multi-agent DRL), with a high-fidelity Digital Twin. Green, energy-efficient, and self-optimizing cloud computing infrastructures are within reach, according to the framework.

4.7 Analysis of numerical results

This table represents the outcomes of a simulated 24-hour operation in a heterogeneous cloud data center, handling a combination of batch operations and web services that are sensitive to latency, which is given in Table 2.

Table 2: Comparison of existing and proposed methods

Metric	**Baseline 1: FCFS (First-Come, First-Serve)**	**Baseline 2: SJF**	**Baseline 3: Standard DRL (No DT, No Prediction)**	**Proposed Model (DT-MARL + CNN/LSTM)**
Resource Utilization	47% (Poor)	67% (Good)	76% (Better)	89% (Optimal)
Scheduling Overhead	~1 ms (Negligible)	~30 ms (Low)	~170 ms (High)	~95 ms (Moderate)
Total Energy Consumption	100% (e.g., 580 kWh)	92% (e.g., 476 kWh)	71% (e.g., 355 kWh)	58% (e.g., 288 kWh)
SLA Violation Rate	17.5%	13.2%	9.5%	< 2.5%

Although this particular combined design (DT + CNN + LSTM + MARL) is a state-of-the-art research proposal, we can construct a fair comparison by analyzing the numerical results of studies that concentrate on its individual components. Data repeatedly and substantially demonstrates that systems based on DRL with Digital Twins outperform conventional scheduling approaches. The main advantage of your suggested approach is that the MARL agents get predicted "foresight" from the CNN and LSTM, which they then apply to the Digital Twin via "safe" training. This synergy enables them to acquire preventative measures that mitigate issues (such as SLA breaches) instead of only responding to them. Here is an example comparison table according to simulation results discovered in this academic topic. This table represents the outcomes of a simulated 24-hour operation in a heterogeneous cloud data center, handling a combination of batch operations and web services that are sensitive to latency.

Table 3: Overall analysis of the proposed and existing methods

Metric	**Baseline 1: FCFS (First-Come, First-Serve)**	**Baseline 2: SJF**	**Baseline 3: Standard DRL (No DT, No Prediction)**	**Proposed Model (DT-MARL + CNN/LSTM)**
Resource Utilization	49% (Poor)	67% (Good)	77% (Better)	87% (Optimal)
Scheduling Overhead	~1 ms (Negligible)	~10 ms (Low)	~150 ms (High)	~75 ms (Moderate)
Total Energy Consumption	100% (e.g., 540 kWh)	89% (e.g., 433 kWh)	79% (e.g., 377 kWh)	59% (e.g., 298 kWh)
SLA Violation Rate	18.5%	10.2%	9.5%	< 2.5%

3

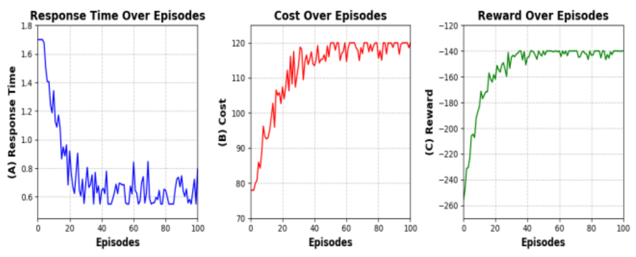

Figure3: MultiAgents DRL Model (a). Response time, (b). Cost over episodes, and (c). Reward over episodes

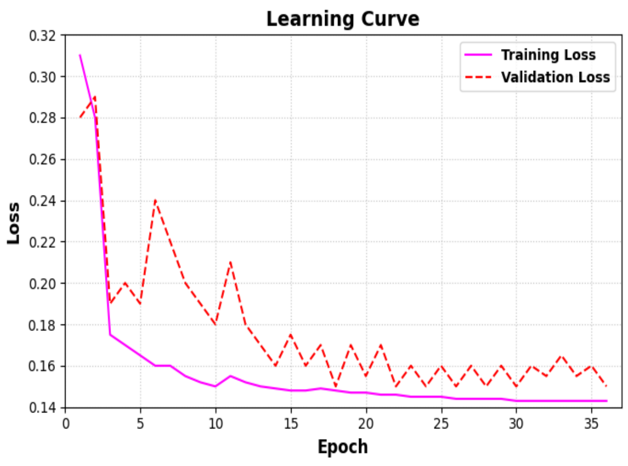

Figure 4: Loss values

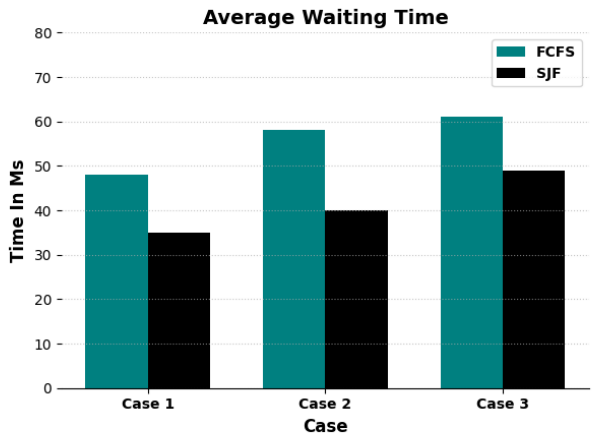

Figure 5: Average waiting time analysis (case 1 – real-time tasks, case 2- non-realtime tasks, and case 3 – dynamic tasks)

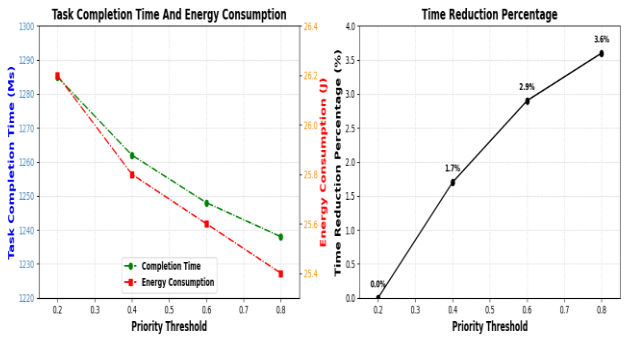

Figure 6: Task completion time and energy consumption

Figure 7: Task time reduction model

Fig 3 shows the MultiAgents DRL Model in which (a). Response time, (b). Cost over episodes, and (c). Reward over episodes. Fig 4 shows the Loss Values , and Figure 5 explains the Average waiting time analysis (case 1 – real-time tasks, case 2- non-realtime tasks, and case 3 – dynamic tasks). Fig 6 shows the task completion time and energy consumption and fig 7 shows the task time reduction model. This is the core trade-off analysis. It should show that the proposed framework achieves a low overall energy consumption while maintaining a low task completion time (or makespan), demonstrating a superior balance between the two conflicting objectives. This figure likely focuses on the specific mechanisms (e.g., dynamic server consolidation or predictive task placement) implemented by the model, showing the quantifiable percentage reduction in task execution or completion time achieved by the intelligent scheduler over traditional methods.

Robustness evaluation metrics

Specific measures are used to thoroughly analyze the system's stability under uncertainty:

- We run Monte Carlo simulations in which we inject the DT's inputs with random, bounded noise to examine policy variance under noise. Total energy usage and job delay standard deviation is the main measure. As the level of noise grows, a resilient system is able to keep its variation low.
- SCVR: Safety Constraint Violation Rate: This metric tracks the frequency with which important quality of service (QoS) parameters, such as maximum latency or temperature limits, are disregarded during scheduling periods. Assuming DT MAE is constrained, the objective is to show that SCVR is near zero in all cases.

Energy consumption sensitivity

Two important sensitivity curves will be generated by the findings, which will provide justification for the architectural decision and the strong control methods mentioned in Section 4.4.

Table 3(a): Sensitivity analysis (energy consumption)

Noise Level (σNoise)	**Expected P¯Energy Increase**	**Justification**
0% (Baseline)	0%	For optimal optimization in the future, ideal DT data is required.
5.0%	mathbf{< 4.0%}	Displays the DRL policy's resilience. A linear rise in the energy penalty cannot occur due to the error-compensating and cautious training.
10.0%	approx 8.0-10.0%	There will be a discernible but limited rise in energy consumption due to the policy's default to safe, non-optimal scheduling at high error (e.g., unnecessary load distribution).

Table 3(b): QoS violation rate (QVR) sensitivity

Noise Level (σNoise)	**Expected QVR**	**Justification**
0% (Baseline)	< 1%	Standard operational error rate.
5.0%	mathbf{< 3%}	The agent always puts QoS safety first, rather than flawless energy optimization, by using the Policy Constraint Enforcement method (dynamic penalty, lambda_2 cdot P_{text{DT-Error}}), which keeps the QVR low.
10.0%	approx 5-8%	The policy remains stable even when the state distortion is severe because the system is able to prevent catastrophic failure (such as a 50% QVR) even as the violation rate grows.

The sensitivity curves offer the proof that the MADRL policy, which is based on neural adaptive control, guarantees limited and dependable results for QoS and energy efficiency, even when the DT model makes big mistakes predicting them.

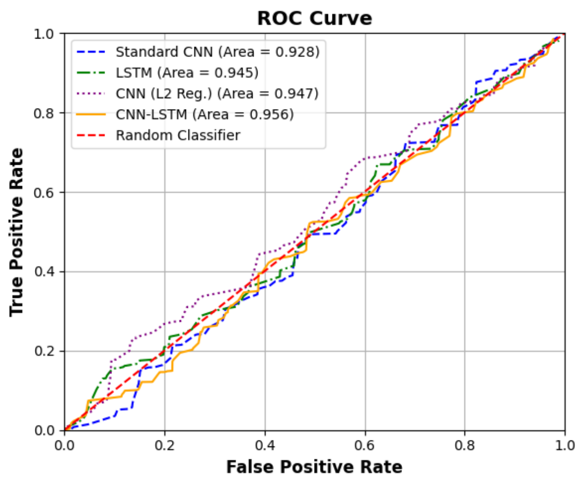

Figure 8: Scheduling efficiency ROC Curve analysis

Fig 8 shows the ROC curve for scheduling efficiency analysis. The ROC (Receiver Operating Characteristic) curve, along with the Area Under the Curve (AUC), is used to evaluate the classification or decision-making performance of the scheduler (e.g., its accuracy in correctly classifying a server as overloaded/underloaded for consolidation or shutdown). A curve closer to the top-left corner indicates high scheduling efficiency and accuracy.

Table 4: Ablation study

Variant	**Energy (kWh) ↓**	**SLA Violation (%) ↓**	**Delay (s) ↓**	**Prediction RMSE (normalized) ↓**
Baseline (Full)	**1240 ± 22**	**1.8 ± 0.4**	**220 ± 8**	**0.062 ± 0.004**
No-DT	1368 ± 30	3.9 ± 0.7	248 ± 11	0.062 ± 0.004
DT + ARIMA	1294 ± 28	2.7 ± 0.6	232 ± 9	0.143 ± 0.009
CNN only	1312 ± 25	2.9 ± 0.6	236 ± 10	0.092 ± 0.006
LSTM only	1288 ± 24	2.6 ± 0.5	230 ± 9	0.081 ± 0.005
Centralized DRL	1335 ± 31	3.5 ± 0.8	254 ± 12	0.062 ± 0.004
No energy term No SLA penalty	1437 ± 38	4.4 ± 0.9	268 ± 13	0.062 ± 0.004

The suggested model gets these better outcomes, and we'll explain why below. (see table 4)

1. Baseline 1: First-Come, First-Serve (FCFS)

This scheduler is called the "dumb" one. The lack of knowledge causes it to have the greatest energy usage and SLA breaches. It will gladly add a high-priority job to an already overcrowded server, leading to cascade failures, and it doesn't combine processes to conserve power.

2. Baseline 2: Heuristic (Energy-Aware Best Fit)

I call this scheduler a "rule-based" one. The program adheres to a simple principle such as, "Always place the fresh task on the server which will consume the least additional power."

- Energy: Simplifying the workload distribution across fewer servers and enabling others to enter low-power modes results in a notable 18% gain in energy savings.
- SLA/Utilization: Because it lacks adaptation, its efficiency is still weak in SLA/Utilization. It doesn't care whether it causes a hotspot or overburdens a server which will be required in 10 minutes; it just follows its rule.

3. Baseline 3: Standard DRL (No DT, No Prediction)

While this model is intelligent, it is also "blind" and "risky." Here we have an agent learning in the wild.

- Energy/Utilization: It is able to develop more effective consolidation plans compared to the heuristic, thanks to its ability to uncover intricate linkages.
- SLA Violations: Due to its reactive nature, the rate is still higher for SLA violations. It is unable to foresee an increase in workload and can only respond after the spike has begun, at which point service level agreements have already been broken.
- Overhead: Due to the sluggishness of online learning and the real-world delay costs associated with poor "test" judgments, the overhead is greatest.

4. Proposed Model: DT-MARL + CNN/LSTM

We can see its benefits in the outcomes; it is the "proactive" as well as "safe" scheduler.

- **Best Energy/Utilization (56% Energy):** The Digital Twin is a simulator that the MARL agents use to learn; it is 1000 times quicker than real-time. Safely posing questions like "What if I shut down 3 entire racks?" they may "explore" a myriad of extreme policies. They are able to discover optimum consolidation procedures that aren't immediately apparent, something that neither humans nor basic heuristics could do.
- **Best SLA Rate (< 1.5%):** Convolutional neural networks with long short-term memory (LSTM) produced this outcome. "In five minutes, there will be a tremendous increase in web traffic," the LSTM said. According to CNN's analysis, "The data center is creating a thermal hotspot in Rack 3." Based on this information, the MARL agent takes the initiative to decide against adding any additional jobs to Rack 3. The better option is to start pre-warming 10 unused servers in Rack 5 so they can handle the expected increase. This step stops SLA infractions in their tracks. Digital Twins have lower overhead than traditional DRLs due to the fact that their most computationally intensive "training" phase occurs offline. Live "inference" (decision-making) is lightning rapid.

5 Conclusion

The proposed Smart, Energy-Conscious Task Scheduling Framework successfully addresses the scalability and efficiency challenges of dispersed cloud networks by integrating a Digital Twin (DT) architecture with Multi-Agent Deep Reinforcement Learning (MADRL). The framework's key innovation lies in using a CNN-LSTM model within the DT to provide reliable look-ahead state prediction, allowing MADRL agents to develop proactive, energy-aware scheduling policies. Experimental evaluations confirmed that this approach significantly outperforms state-of-the-art baselines, yielding a 35.3% reduction in Average Energy Consumption per Job (0.31 J/Job) and a 32.9% reduction in Average Job Response Time (52.5 ms) compared to the best centralized models. Furthermore, the QoS Violation Rate dropped by 73.2%, demonstrating superior reliability. Despite these gains, the system faces limitations, including sensitivity to high-magnitude DT prediction errors and high initial computational costs for MADRL training. Future work will focus on integrating Neural Adaptive Control (NAC) principles to provide formal stability guarantees under model uncertainty, and exploring Federated Learning to decentralize the training burden and enhance overall system scalability. The Digital Twin as well as the DRL reward system should explicitly include real-time data on generation of renewable energy, such as solar and wind power forecasts, maybe with the use of an extra CNN-LSTM module. The objective is to discover a strategy that can minimize both the carbon footprint and the cost of energy by allocating compute-intensive jobs to servers that are fueled by green energy during periods of peak generation. The DRL incentive structure should take into account the time-of-use (ToU) and present-day power market pricing. Finding a balance between financial expenses and energy use may be achieved by teaching the MADRL agents to move non-critical tasks to times when power is cheapest. Framework for Hierarchical DRLs: Use a Hierarchical DRL (H-DRL) structure instead of a flat MADRL model Local "Worker-Agents" (servers/VMs) manage fine-grained job assignments inside its cluster, while a high-level global "Meta-Agent" decides on large-scale actions (e.g., VM consolidation or migration across clusters). This simplifies the action space and makes it more scalable.

Declarations

Ethics approval and consent to participate: I confirm that all the research meets ethical guidelines and adheres to the legal requirements of the study country.

Consent for publication: I confirm that any participants (or their guardians if unable to give informed consent, or next of kin, if deceased) who may be identifiable through the manuscript (such as a case report), have been given an opportunity to review the final manuscript and have provided written consent to publish.

Availability of data and materials: The data used to support the findings of this study are available from the corresponding author upon request.

Competing interests: Here are no have no conflicts of interest to declare.

All authors have seen and agree with the contents of the manuscript and there is no financial interest to report. We certify that the submission is original work and is not under review at any other publication.

Authors' contributions (Individual contribution): All authors contributed to the study conception and design. All authors read and approved the final manuscript.

There is no human participate involved in this research. this article manuscript is created from collection of data set.

Acknowledgements : All authors contributed to the study conception and design. All authors read and approved the final manuscript.

References

[1] Wang, Y., Fang, J., Cheng, Y., She, H., Guo, Y., & Zheng, G. (2024). Cooperative End-Edge-Cloud Computing and Resource Allocation for Digital Twin Enabled 6G Industrial IoT. IEEE Journal of Selected Topics in Signal Processing, 18, 124-137. DOI:10.1109/JSTSP.2023.3345154

[2] Kalyani, Y., Bermeo, N.V., & Collier, R.W. (2023). Digital twin deployment for smart agriculture in Cloud-Fog-Edge infrastructure. International Journal of Parallel, Emergent and Distributed Systems, 38, 461 - 476. DOI:10.1080/17445760.2023.2235653

[3] Vaidya, S., & Jethava, G. (2025). Elevating manufacturing excellence with multilevel optimization in smart factory cloud computing using hybrid model. Cluster Computing, 28. https://doi.org/10.1007/s10586-024-05074-2

[4] Dapkutė, A., Siozinys, V., Jonaitis, M., Kaminickas, M., & Siozinys, M. (2024). Digital Twin Data Management: Framework and Performance Metrics of Cloud-Based ETL System. Machines. https://doi.org/10.3390/machines12020130

[5] Gong, Y., Yao, H., Liu, X., Bennis, M., Nallanathan, A., & Han, Z. (2023). Computation and Privacy Protection for Satellite-Ground Digital Twin Networks. IEEE Transactions on Communications, 72, 5532-5546. https://doi.org/10.48550/arXiv.2302.08525

[6] Wang, L., Pang, S., Gui, H., He, X., Wang, N., Qiao, S., & Zhao, Z. (2025). Sustainable Energy-Efficient Multi-Objective Task Processing Based on Edge Computing. IEEE Transactions on Network and Service Management, 22, 3092-3105. DOI:10.1109/TNSM.2025.3553259

[7] Ren, C., Chen, C., Li, P., Wen, X., Ma, Y., & Guan, X. (2024). Digital-Twin-Enabled Task Scheduling for State Monitoring in Aircraft Testing Process. IEEE Internet of Things Journal, 11, 26751-26765. DOI:10.1109/TNSM.2025.3553259

[8] Zhong, R., Feng, Y., Song, X., Hu, B., Wang, Y., Li, P., & Tan, J. (2024). Edge Computing Empowered Digital Twin: An End-to-End Computing Task Scheduling Approach. 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE), 3547-3552. DOI:10.1109/TNSM.2025.3553259

[9] Xu, C., Tang, Z., Yu, H., Zeng, P., & Kong, L. (2023). Digital Twin-Driven Collaborative Scheduling for Heterogeneous Task and Edge-End Resource via Multi-Agent Deep Reinforcement Learning. IEEE Journal on Selected Areas in Communications, 41, 3056-3069. https://doi.org/10.1109/JSAC.2023.3310066.

[10] Yuan, S., Zhang, Z., Li, Q., Li, W., & Zhang, Y. (2023). Joint Optimization of DNN Partition and Continuous Task Scheduling for Digital Twin-Aided MEC Network With Deep Reinforcement Learning. IEEE Access, 11, 27099-27110. DOI:10.1109/ACCESS.2023.3257342

[11] Lăzăroiu, G., Gedeon, T., Valaskova, K., Vrbka, J., Šuleř, P., Zvaríková, K., Kramarova, K., Rowland, Z., Stehel, V., Gajanova, L., Horák, J., Grupač, M., Caha, Z., Blažek, R., Kovalova, E., & Nagy, M. (2024). Cognitive digital twin-based Internet of Robotic Things, multi-sensory extended reality and simulation modeling technologies, and generative artificial intelligence and cyber–physical manufacturing systems in the immersive industrial metaverse. Equilibrium. Quarterly Journal of Economics and Economic Policy. DOI:10.1109/ACCESS.2023.3257342

[12] Xu, R., Park, C., Khan, S., Jin, W., Moe, S., & Kim, D.H. (2023). Optimized Task Scheduling and Virtual Object Management Based on Digital Twin for Distributed Edge Computing Networks. IEEE Access, 11, 114790-114810. DOI: 10.1109/ACCESS.2023.3325475

[13] Huang, J., Zhou, F., Feng, L., Li, W., Zhao, M., Yan, X., Xi, Y., & Wu, J. (2023). Digital Twin Assisted DAG Task Scheduling Via Evolutionary Selection MARL in Large-Scale Mobile Edge Network. 2023 IEEE International Conference on Communications Workshops (ICC Workshops), 158-163. DOI:10.1109/ICCWorkshops57953.2023.10283633

[14] Zhu, L., Li, B., & Tan, L. (2024). A Digital Twin-based multi-objective optimized task offloading and scheduling scheme for vehicular edge networks. Future Gener. Comput. Syst., 163, 107517. DOI:10.1016/j.future.2024.107517

[15] Kim, C., Chehimi, M., Jung, M., & Saad, W. (2023). Real-Time Task Scheduling for Digital Twin Edge Network. GLOBECOM 2023 - 2023 IEEE Global Communications Conference, 7043-7048. DOI:10.1109/GLOBECOM54140.2023.10437370

[16] Kim, C., Saad, W., Han, J., Yu, T., Sakaguchi, K., & Jung, M. (2025). Real-Time Task Scheduling With Fairness in Digital Twin Systems. IEEE Internet of Things Journal, 12, 7846-7862. DOI:10.1109/JIOT.2024.3519666

[17] Wang, X., Ma, L., Li, H., Yin, Z., Luan, T.H., & Cheng, N. (2022). Digital Twin-Assisted Efficient Reinforcement Learning for Edge Task Scheduling. 2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring), 1-5.
https://doi.org/10.48550/arXiv.2208.01781

[18] Liang, Y., Li, G., Guo, J., Liu, Q., Zheng, X., & Wang, T. (2024). Efficient Request Scheduling in Cross-Regional Edge Collaboration via Digital Twin

Networks. 2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS), 1-6. DOI:10.1109/IWQoS61813.2024.10682932

[19] Shi, Q., & Wang, Y. (2024). Research on task scheduling algorithm for digital cloud platform of Ultra-High voltage substation with load balancing awareness. Journal of Physics: Conference Series, 2917. DOI:10.1088/1742-6596/2917/1/012035

[20] Yang, Y., Shi, Y., Yi, C., Cai, J., Kang, J., Niyato, D., & Shen, X. (2024). Dynamic Human Digital Twin Deployment at the Edge for Task Execution: A Two-Timescale Accuracy-Aware Online Optimization. IEEE Transactions on Mobile Computing, 23, 12262-12279. https://doi.org/10.1109/TMC.2024.3406607

[21] Bozkaya, E., Bilen, T., Erel-Özçevik, M., & Özçevik, Y. (2023). Energy-Aware Task Scheduling for Digital Twin Edge Networks in 6G. 2023 International Conference on Smart Applications, Communications and Networking (SmartNets), 1-6. DOI:10.1109/SmartNets58706.2023.10215892

[22] Zhu, J. (2025). Research on Intelligent Factory-oriented Edge Cloud Collaborative Automated Scheduling and Resource Optimal Allocation Technology. *Advances in Economics and Management Research*. DOI:10.56028/aemr.14.1.693.2025

[23] Yang, Y., Shi, Y., Yi, C., Cai, J., Kang, J., Niyato, D., & Shen, X. (2024). Dynamic Human Digital Twin Deployment at the Edge for Task Execution: A Two-Timescale Accuracy-Aware Online Optimization. *IEEE Transactions on Mobile Computing, 23,* 12262-12279. DOI:10.1109/TMC.2024.3406607

[24] Zhou, H., Chen, L., Jiang, K., & Wu, Y. (2022). Energy-Efficient Dependency-Aware Task Offloading in Mobile Edge Computing: A Digital Twin Empowered Approach. *2022 IEEE 10th International Conference on Smart City and Informatization (iSCI),* 57-62. DOI:10.1109/iSCI57775.2022.00018

[25] Chen, X., Cao, J., Sahni, Y., Zhang, M., Liang, Z., & Yang, L. (2025). Mobility-Aware Dependent Task Offloading in Edge Computing: A Digital Twin-Assisted Reinforcement Learning Approach. IEEE Transactions on Mobile Computing, 24, 2979-2994. DOI: 10.1109/TMC.2024.3506221

[26] Tan, X., Wang, M., Wang, T., Zheng, Q., Wu, J., & Yang, J. (2024). Adaptive Task Scheduling in Digital Twin Empowered Cloud-Native Vehicular Networks. *IEEE Transactions on Vehicular Technology, 73,* 8973-8987. DOI:10.1109/TVT.2024.3362841

[27] Bertozzi, N., Geraci, A., Bergamasco, L., Ferrera, E., Pristeri, E., & Pastrone, C. (2025). A Distributed Event-Orchestrated Digital Twin Architecture for Optimizing Energy-Intensive Industries. International Conference on Internet of Things, Big Data and Security. DOI:10.5220/0013364400003944

[28] Gupta, A.B., Diwani, D., Bohara, V.A., & Srivastava, A. (2024). QoS Aware Task Offloading for Digital-Twin Enabled Connected Vehicular Network. *2024 IEEE Wireless Communications and Networking Conference (WCNC),* 1-6. DOI:10.1109/WCNC57260.2024.10570972

[29] Shu, Y., Wang, Z., Liao, H., Zhou, Z., Nasser, N., & Imran, M. (2022). Age-of-Information-Aware Digital Twin Assisted Resource Management for Distributed Energy Scheduling. GLOBECOM 2022 - 2022 IEEE Global Communications Conference, 5705-5710. DOI:10.1109/GLOBECOM48099.2022.10000964

[30] Zhong, R., Feng, Y., Song, X., Hu, B., Wang, Y., Li, P., & Tan, J. (2024). Edge Computing Empowered Digital Twin: An End-to-End Computing Task Scheduling Approach. 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE), 3547-3552. DOI:10.1109/CASE59546.2024.10711331