JCTA Journal of Computing Theories and Applications ISSN:3024-9104

Research Article

Hybrid Dynamic Programming Healthcare Cloud-Based Quality of Service Optimization

Nengak I. Sitlong [1,*], Abraham E. Evwiekpaefe [2], and Martins E. Irhebhude [2]

[1] Faculty of Sciences, Computer Science Department, Federal University of Education (FUEP), Pankshin, Plateau State 930001, Nigeria; e-mail : iliya_sitlong@yahoo.com
[2] Faculty of Military Science and Interdisciplinary Studies, Computer Science Department, Nigerian Defence Academy (NDA), Kaduna 700001, Nigeria; e-mail : aeevwiekpaefe@nda.edu.ng; mirhebhude@nda.edu.ng
* Corresponding Author : Nengak I. Sitlong

Abstract: The integration of Internet of Things (IoT) with cloud computing has revolutionized healthcare systems, offering scalable and real-time patient monitoring. However, optimizing response times and energy consumption remains crucial for efficient healthcare delivery. This research evaluates various algorithmic approaches for workload migration and resource management within IoT cloud-based healthcare systems. The performance of the implemented algorithm in this research, Hybrid Dynamic Programming and Long Short-Term Memory (Hybrid DP+LSTM), was analyzed against other six key algorithms, namely Gradient Optimization with Back Propagation to Input (GOBI), Deep Reinforcement Learning (DRL), improved GOBI (GOBI2), Predictive Offloading for Network Devices (POND), Mixed Integer Linear Programming (MILP), and Genetic Algorithm (GA) based on their average response time and energy consumption. Hybrid DP+LSTM achieves the lowest response time (82.91ms) with an energy consumption of 2,835,048 joules per container. The outcome of the analysis showed that Hybrid DP+LSTM have significant response times improvement, with percentage increases of 89.3%, 79.0%, 83.8%, 97.0%, 99.8%, and 99.94% against GOBI, GOBI2, DRL, POND, MILP, and GA, respectively. In terms of energy consumption, Hybrid DP+LSTM outperforms other approaches, with GOBI2 (3,664,337 joules) consuming 29.3% more energy, DRL (2,973,238 joules) consuming 4.9% more, GOBI (4,463,010 joules) consuming 57.4% more, POND (3,310,966 joules) consuming 16.8% more, MILP (3,005,498 joules) consuming 6.0% more, and the GA (3,959,935 joules) consuming 39.7% more. The result of ablation of the Hybrid DP+LSTM model achieves a 47.05% improvement over DP-only (156.57ms) and a 70.64% improvement over LSTM-only (282.41ms) in response time. On the energy efficiency side, Hybrid DP+LSTM shows 22.80% improvement over LSTM-only (3,671,51 joules), but 7.34% underperformance compared to DP-only (2,640,93). These research findings indicate that the Hybrid DP+LSTM technique provides the best trade-off between response time and energy efficiency. Future research should further explore hybrid approaches to optimize these metrics in IoT cloud-based healthcare systems.

Keywords: Cloud; Dynamic Programming; Energy Optimization; Fog/Edge Computing; Healthcare; IoT; LSTM; Task Scheduling.

Received: August, 25th 2025
Revised: September, 20th 2025
Accepted: September, 23rd 2025
Published: September, 26th 2025

1. Introduction

As the Internet of Medical Things (IoMT) proliferates within healthcare, challenges around latency and responsiveness have become increasingly critical, particularly in applications such as wearable vital sign monitoring, remote surgery, and real-time diagnostics [1]. Traditional IoT-to-cloud architectures often fail to meet stringent timing requirements due to network delays, which has driven the shift toward fog/edge architectures and AI-based scheduling for latency minimization [2]. The constraint with the deployment of fog/edge architecture is the competition for scarce fog/edge resources among numerous IoT tasks on the fog layer. Task offloading and load balancing remain the key strategies for resolving the scarce resource constraint in the fog/edge layer. A formalized queuing and delay distribution

model for three-tier IoT–fog–cloud systems was implemented by optimizing offloading probabilities via a sub-gradient-based algorithm that minimizes the probability a task exceeds its latency threshold [3]. Furthermore, Transformer PPO Task Offloading (TPTO), employs a Transformer + PPO deep reinforcement learning model to schedule dependent tasks on edge servers. TPTO achieves 30% lower latency than baseline heuristics (e.g., HEFT, Greedy) and converges 2.5 times faster than traditional DRL techniques [4].

Metaheuristic scheduling approaches also play a significant role in optimizing latency in IoT cloud-based systems. A Lightweight Secure Efficient Offloading Scheduling (LSEOS) algorithm, leveraged on neighborhood search and adaptive deadlines to reduce latency and overhead in fog environments [5]. Edge AI strategies such as federated learning (FedAvg, FedDyn, Sub FedAvg) enable distributed model training close to IoMT devices, minimizing latency by keeping inference local and reducing communication overhead [6]. On-device optimization techniques, such as pruning, quantization, and parallelization scheduling, are used to reduce model size and accelerate inference on constrained edge hardware [1]. The Smart Edge architecture unites ensemble machine learning models (voting-based classifiers) deployed across the edge and cloud to predict diabetes risk in real-time. Ensemble voting enhances prediction accuracy by ~5% while achieving low latency over a hybrid infrastructure [7]. Moreover, a technique called a hybrid CNN LSTM DL model was deployed at edge nodes, communicating over 5G Ultra-Reliable Low-Latency Communication (URLLC) links to support real-time patient monitoring with end-to-end latencies at 96.5% accuracy, nearly a 47% speedup over federated learning techniques [2]. Complementing ML strategies, traditional measures such as window-based rate control algorithms (WRCAs) have been utilized in telesurgery and medical edge systems. These methods optimise for mean latency, jitter, and peak-to-mean ratios to satisfy medical QoS constraints, outperforming other rate control techniques, such as Battery Smoothing algorithms [8].

Furthermore, the Gradient-Based Optimization with Back Propagation to Input (GOBI) algorithm [9] produced one of the best IoT cloud QoS with improved energy consumption and reduced latency. This was achieved by feeding back the same result from a surrogate neural network model to itself through a Monte Carlo simulation, thereby eliminating the problem of local optima. The process converged after an undefined number of iterations because the confidence level of the best response time is unknown [10]. The deficiency here is that feeding the same data repeatedly into the same machine-learning algorithm does not necessarily improve its accuracy; rather, it results in neural network data saturation, which increases computational complexity and inefficiency by solving the same computation repeatedly—a situation described as "Overlapping Sub-Problems" [11]. In addition, the standard medically guaranteed safety response time margin is set at 150 to 400 milliseconds [12]. However, the optimized IoT response time achieved by Tuli et al. (2021) was not within this range as their result was in seconds and not milliseconds.

Medically, 10 ms is the standard response time approved for remote surgery procedures and remote patient monitoring [13]. Previous optimization efforts, such as the MILP model, achieved a 73% improvement compared to cloud-only IoT response times [14]. Building on this, the present research aims to further enhance quality of service—specifically response time and energy consumption—by proposing a hybrid model that combines Dynamic Programming (DP) for analytical scheduling with Long Short-Term Memory (LSTM) for predictive learning, deployed via transfer learning.

It should be noted that the evaluation of the proposed IoT Healthcare Fog–Cloud model is conducted under simulated conditions. The experiments utilize the COSCO platform and Bitbrains traces as benchmark datasets. While Bitbrains offers realistic cloud workload patterns, the findings do not constitute direct testing in real healthcare infrastructure but rather controlled simulations designed to approximate realistic operating conditions.

2. Related Works

A standard design for cloud-based IoT healthcare systems should encompass key factors, such as reliability, scalability, privacy, security, quality of service (QoS), and data storage, as core implementation goals in developing the architecture of an IoT network [15]. Crypto-security algorithms, such as the RSA encryption algorithm and Digital Signature Standard (DSS), were utilized in all endpoints of IoT networks to provide security [16]. However, this system is deficient in the speed of the feedback mechanism due to the design complexities of

the machine learning algorithm, which is deployed far from the edge computing side of the system in the cloud. In an effort to minimize this effect, wearable devices were empowered to perform processing and visualization of sensor data directly, providing prompt feedback to the user. However, this poses an energy overhead on the processing IoT wearable device, which is sustained by a battery power source [17].

The wearables were regarded as having low memory and processing capabilities, and the healthcare data emanating from the sensors are better transferred and stored on a database server, and subsequently extracted by medical personnel through the use of a mobile application for medical advice [18]. This design utilized a Raspberry Pi to intercept data generated by IoT sensors, such as temperature or ECG, and transfer it to the database server interface via a mobile application, providing data access to healthcare advisers or medical doctors when needed. The design tends to overlook key IoT performance metrics, which determine QoS, such as security, latency, energy consumption, and bandwidth, as it focuses solely on providing adequate storage space on the database server. The security metrics were improved through the implementation of a Message Queue Telemetry Transport (MQTT) broker, which publishes ECG data to the web server [19].

Furthermore, a secure healthcare monitoring system that integrates Named Data Networking (NDN)- based IoT with the edge cloud has been proposed [20]. The system leverages the benefits of NDN to expedite the retrieval of medical data and utilizes cyphers and signatures to ensure that data is delivered securely. Quantitative analysis of the results indicates that the system reduces medical data retrieval delay and cost by 28% and 52%, respectively. The method was modified to include methods for protecting data through encryption and secure key exchange, which authenticate patients using effective hashing of selected data [21]. This enables privacy protection while also considering the network's irregular occurrences by providing redundancy to prevent data loss. A distributed platform, such as the cloud, offers limitless computing resources and a wide range of services. However, utilizing these services with minimum latency and delay is not guaranteed [22]. One of the prime components in achieving optimal performance is ensuring high bandwidth for data transmission. To minimize delay, a middleware layer known as "fog computing" is to be placed between Cloud and IoT, the "fog computing" will achieve low latency for applications that are sensitive to delay, such as the health sector [23]. As sensor nodes have constrained power backup, the design of the fog-computing layer should be computationally light and consume minimal energy. To conserve energy, the solution could be to exploit alternate forms of energy generation models as well as use efficient techniques to program periodic sleep routines for the sensor nodes [24]. The devices can enter a sleep mode if no observable sensing activity occurs within a given period. End-to-end connectivity, low latency, and reliability-enhanced solutions in IoT enable the simultaneous monitoring of vulnerable patients without requiring dedicated healthcare staff [25].

Emerging machine learning technologies in wireless communications is delivering a great service [26]. Reinforcement learning is a branch of machine learning in which an agent selects an action from a provided list of actions. With every action, the agent receives a certain reward, and depending on the action, it can be a positive or negative reward [27]. The agent takes certain actions in the environment and learns from the consequences of those actions. Reinforcement learning IoT based communication framework for healthcare applications was used to check resource utilization for improve QoS, under this, IoT tasks generated were passed through a neural network algorithm which assigns reward to the tasks based on the critical state of such tasks, the task with the most critical state was prioritized and resources allocated for required action to be taken, and consequently improved the IoT response time of the healthcare system [28]. Similarly, the Median Attribute Deviation (MAD) of the CPU utilization of the hosts was estimated, and select containers based on the Maximum Correlation (MC) with other tasks [29]. However, such heuristic-based approaches fail to model healthcare environments with dynamic workloads [30].

Evolutionary approaches like GA lie in the domain of gradient-free optimization methods. The GA method schedules workloads using a neural model to approximate the objective value and a GA to reach the optimal decision. Moreover, nonlocal jumps can lead to a significant change in the scheduling decision, resulting in a high number of migrations [31]. This effect was improved through Predictive Offloading for Network Devices (POND), a variant of the Max-Weight approach [32]. POND uses constrained online dispatch with unknown

arrival and reward distributions [33]. The final decision is made using the Max-Weight approach, where the weights correspond to the expected reward values. To minimize objective value, an objective score was provided as a negative reward [31]. Due to the inability of Max-Weight approaches to adapt to volatile scenarios, their wait times are high.

Reinforcement learning-based methods have demonstrated robustness and versatility in handling diverse workload characteristics and complex edge setups [34]. Such methods employ a Markovian assumption of the state, which is the scheduling decision at each interval. Based on new observations of the reward signal, this explores its knowledge of the state space to converge to an optimal decision [10]. The result obtained showed minimal improvement in IoT response time. This research was enhanced to achieve a minimal latency of 30 to 50% in IoT cloud response time through the use of a Greedy algorithm and Fuzzy Inference Reinforcement Learning (FIRL) [12]. The COSCO framework was used to implement GOBI, and later enhanced with Monte Carlo simulation, which fed the output of a surrogate neural network model back to itself to achieve GOBI*, thereby preventing the tendency to get stuck at local optima when finding the optimal response time of the cloud [9]. This was developed to leverage a seamless interface between the orchestration framework and simulation engine, providing an interactive dynamic interface between AI models and resource metrics to optimise QoS.

To meet the needs of IoT healthcare applications, it is necessary to maximize the use of computing layers, such as edge, fog, and cloud layers. These layers offer various computing capabilities, including processing, storage, and internet-based communication interfaces [35]. Consequently, scalable data analytics and reliable solutions to address the challenges of IoT healthcare applications, such as reducing service execution time and energy consumption, were provided through the use of an integrated edge-fog-cloud architecture [19]. When it comes to placing IoT healthcare applications optimally, the emphasis is on the significance of edge, fog, and cloud, while considering several performance metrics, including energy consumption, service latency, resource usage, and security [36]. An integrated edge-fog-cloud healthcare architecture was developed to reduce the energy consumption and service latency of IoT healthcare applications. The results showed that energy consumption and latency were reduced by 27% and 28%, respectively [22]. Furthermore, an efficient framework based on the Remora Optimization algorithm was created to jointly minimize latency and energy consumption in an IoT healthcare system. The framework was considered efficient in improving the energy consumption of IoT devices and reducing the response time delay of IoT sensors [37].

To reduce IoT device energy consumption and delay, a novel message exchange procedure with load balancing was also presented to offload IoT healthcare applications via a cloud-fog architecture [38]. The research reduced energy usage and delay by 77% and 60%, respectively, through the deployment of an energy-efficient fuzzy technique. Additionally, the use of GA to offer a safe and economical way to utilize a Wireless Sensor Network (WSN) for the detection of infectious diseases was implemented to mitigate the transmission rate from one victim to another through early detection and treatment [35]. An optimization model called the Mixed-Integer Non-Linear Programming (MINLP) model was implemented using improved deep reinforcement learning (DRL) technique to determine the best course of action for allocating resources, offloading computation, and reducing energy usage of cloud-based IoT healthcare system and the result showed better resource management at the fog computing layer that translated to minimal IoT device energy usage [39]. Furthermore, a hybrid energy-efficient model for patient home monitoring was proposed, which utilized sensors to record and analyze electrocardiograms (ECGs). The results showed lower energy usage, latency, and network utilization, as observed [40].

A model for an efficient cloud-based IoT vehicle routing problem using a resource management algorithm called Reliable Resource Allocation Management (R2AM) was implemented on the iFogSim2 simulator, featuring 15 fog devices and an installed Decentralized Decision System (DDS) for resource management [41]. The IoT response time and energy consumption obtained from the R2AM algorithm, as compared to the result of one of the existing algorithms, called Scalable Microservices Placement (SMP), showed that R2AM has 10.3% lower latency and 21.85% lower energy usage than the SMP algorithm. However, R2AM has limitations due to execution failure, which is caused by the limited number of fog devices used [41]. In order to further optimize cloud based IoT energy and service latency QoS, a four-tier architecture consisting of Internet of Medical Things (IoMT) layer, edge layer,

fog layer, and cloud layer was presented, this was harness by categorizing the IoT generated workloads into high critical sensor data (such as heart failure rate, body temperature, glucose level rating), medium critical sensor data (such as remote patient monitoring, emergency response service, patient diagnostic, health status check), and low critical sensor data (Example, healthcare records requiring complex data analytics for future decision making), these data were offloaded into the setup using MILP resource management model to optimize service level objective matric specifically energy consumption and service latency of IoT cloud healthcare solution, the result obtained from the four-tier MILP model was compared against that obtained from cloud only architecture (described as baseline architecture), and the result showed that the energy consumption and service latency were optimized by 21% and 73%, respectively [14]. Collectively, the trajectory from 2023 through early 2025 shows a clear evolution from heuristic or basic offloading architectures toward hybrid, AI-driven scheduling systems, reinforcement learning, and deep learning models co deployed across edge and cloud. These frameworks not only minimize latency but also guarantee predictable QoS even under high load, dynamic workloads, and in latency critical healthcare contexts.

3. Proposed Method

The research method is centered on the heterogeneous cloud-based IoT model in Figure 1, comprising the IoT layer, the Fog layer and the cloud layer. IoT response time, as well as other Service Level Objectives (SLOs), is enhanced by placing numerous Fog or Edge layers closer to users for caching data and providing fast access to cloud resources. IoT devices exhibit lower latency when accessing data from the Fog layer through the fog gateway, which is faster than accessing it directly from a higher-latency cloud layer that is remotely situated. However, the hosts at the fog layer have lesser computational capabilities relative to the hosts on the cloud layer. The functionalities performed at the fog layer comprise the scheduling decision process, data caching, resource evaluation schemes, load balancing, and task offloading, which enable necessary trade-offs between the fog and cloud layers to improve task efficiency.

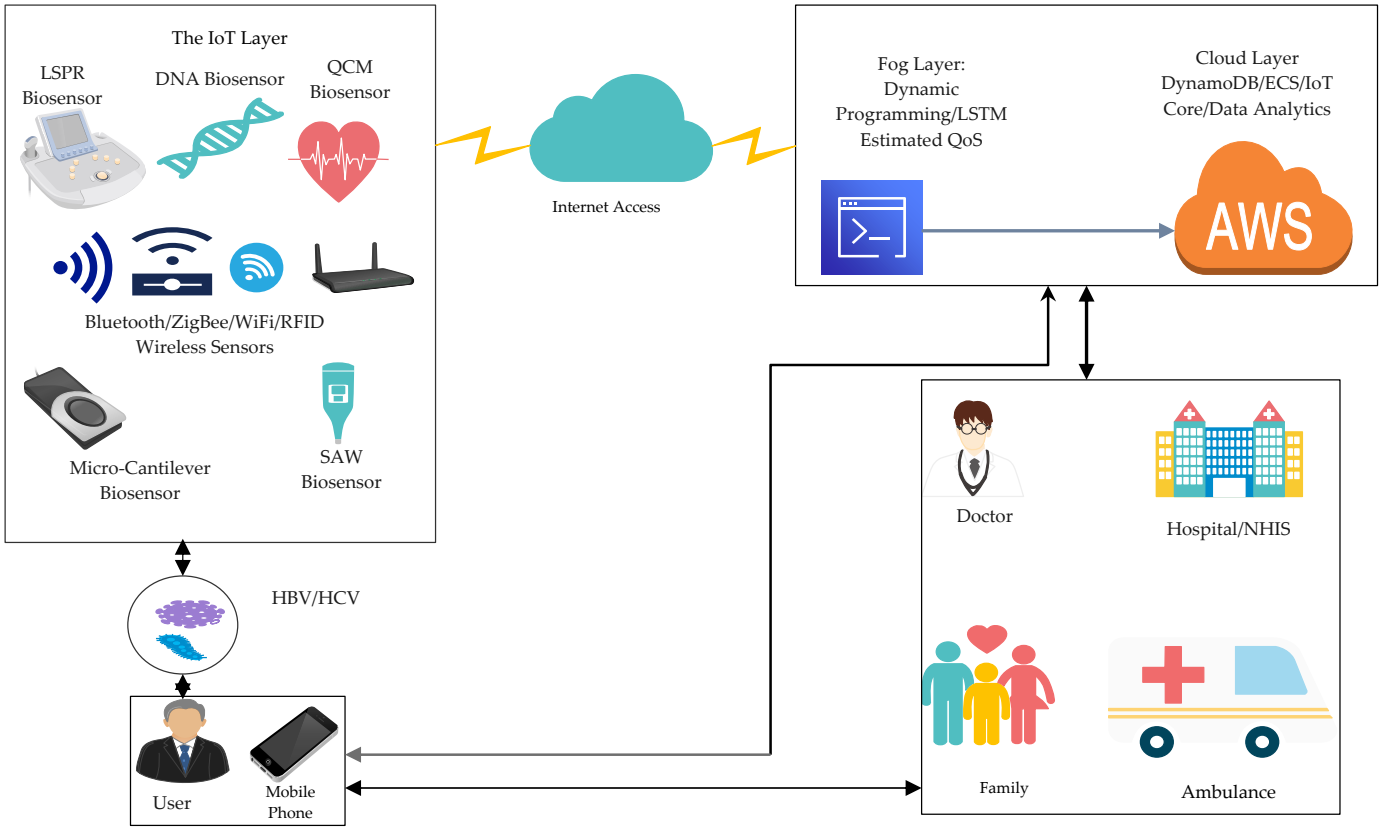

Figure 1. Proposed Cloud-Based IoT Healthcare Solution

The IoT layer is where sensors and user devices are located in the cloud-based IoT model. In this context, the sensors are streamlined to those responsible for non-invasive interception and transmission of signals related to Hepatitis. Specifically, the sensors of interest to this research, as described, are Localized Surface Plasmon Resonance (LSPR), DNA Biosensor, Quartz Crystal Microbalance (QCM), Microcantilever Biosensor, and Surface Acoustic Wave (SAW) Biosensor. All these sensors will be simulated using JSON codes and will be responsible for generating the workloads to be communicated to the fog layer through the fog gateway device. The workloads will then be orchestrated into containers known as tasks to be executed on the fog broker. Consequently, the fog broker handles task execution by making the best scheduling decision based on the data forwarded by the sensors and allocating tasks appropriately to hosts on the fog layer, taking into account resource metrics such as CPU, RAM, Disk, and Network bandwidth.

The objective of this research is to implement a hybrid DP task scheduler on the fog layer, which will optimize the response time and energy consumption achieved by the MILP model. This is intended to be achieved by implementing an interface between the container orchestration and resource metric monitoring service, utilizing an enhanced DP algorithm to optimize the communication response time of IoT devices, as well as other SLO metrics. The total number of hosts VM at the fog layer will be designated as $VM = \{vm_0, vm_1, \ldots, vm_{n-1}\}$ and the time series optimization metrics for CPU, RAM, Disk, and Bandwidth usage of host vm will be U(vm_i^t). Furthermore, the set of total capacities of CPU, RAM, Disk, and Bandwidth with the mean response latency of host vm_i will be designated as $C(vm_i)$.

3.1. IoT Sensors Tasks Model

The workload for the model originates from the IoT layer, where the total time slice for sensor task creation is divided into uniform scheduling intervals of duration d as illustrated in Figure 2. The K-th interval is represented as T_k with start point of $s(T_k)$, such that $s(T_0) = 0$, and $s(T_k) = s(T_{k-1}) + \mathrm{d}$ for every $K > 0$. At time interval $s(T_{k-1})$, a total of N_k new tasks are generated by IoT devices. These tasks, along with their SLOs—such as Instruction Per Second (IPS), RAM, disk, and bandwidth requirements—are transmitted to the fog layer for processing. The set of active tasks at time interval T_k is given by $W_k = \{w_0^k, w_1^k, \ldots, w_{|W_k|}^k\}$, where w_i^k denotes the i-th task within the list of tasks.

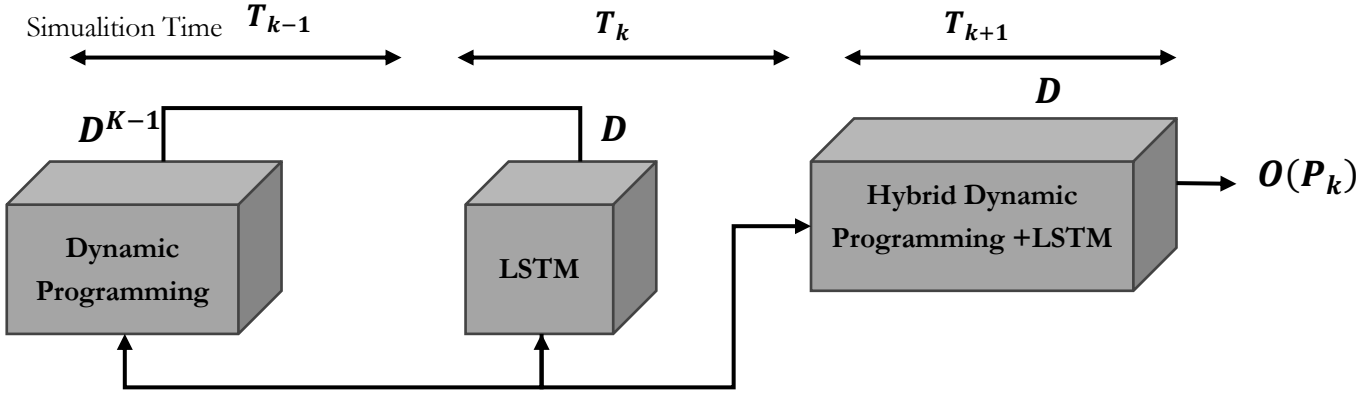

Figure 2. Proposed hybrid model

The fog layer schedules the newly generated tasks using DP, as formulated in Equation (10) in Section 3.3, on the EC2 compute host. The resulting scheduling decision is represented as $D_k = \{(h, \tau)\}$, where D_k is a list of ordered pairs consisting of host identifiers and container/task identifiers.

At the completion of each interval T_k the set of tasks successfully executed is represented as C_k. The unfinished tasks are then moved to the next interval, forming $W_{k+1} = N_k \cup W_k \cup Q_k$, where N_k is the set of newly generated tasks, W_k is the set of active tasks, and Q_k represents tasks waiting in the queue. In this way, the fog layer scheduling mechanism

determines whether tasks remain active, are completed, or require migration to another host. The combination of DP for deterministic scheduling and LSTM for predictive QoS estimation enables the hybrid approach. The conceptual workflow of this model is illustrated in Figure 2.

3.2. Response Time, Energy Consumption, SLA Violation, Container Destroyed, and Migration Time QoS Models

The primary focus of this research is the optimization of response time and energy consumption, with other metrics serving as supporting indicators. The total response time of IoT devices is the sum of the scheduling time and the task execution time for a given workload [41]. The tasks are organized into containers on the fog broker, and the sensor-generated data are used as input for allocation to hosts with higher resources. Task migration scheduling is implemented within the cloud fog broker to ensure proper task allocation to suitable Virtual Machines (VMs). This decision focuses on optimizing key IoT QoS parameters, specifically response time and energy consumption. Let the objective function, based on the optimal response time and energy consumption within the time interval T_k (Figure 2), be:

$$O(P_k) = \alpha . AEC_k + \beta . ART_k \tag{1}$$

Where AEC = average energy consumption; ART= average response time; $AEC\&ART \in P_k$.

The average energy consumption is computed for any time interval for a device, normalized by the device's maximum power, as shown in Equation (2).

$$AEC_k = \frac{\sum_{vm \in VM} \int_{s(T_k)}^{s(T_{k+1})} Power_{vm}(T) dT}{|W_k| \sum_{vm_k \in VM}^{\sum} Power_{vm}^{max}} \tag{2}$$

Where W_k= set of active tasks at time interval T_k; VM_k = total number of hosts at interval T_k; uniform time scheduling interval of duration.

The average response time is computed for interval T_k for all completed tasks C_k normalized by the maximum response time before the present interval, as shown in Equation (3).

$$ART_k = \frac{\sum_{\tau \in C_k} Re\,s\,ponseTime(\tau^k)}{|C_k|\, max_{s \le k}\, Re\,s\,ponseTime(T_s^j)} \tag{3}$$

The number of containers destroyed refers to the total number of containers destroyed within an interval, as given in Equation (4).

$$numdesroyed_t = |destroyed_t| \tag{4}$$

When tasks are inefficiently allocated, they must be destroyed and re-initiated on another host with adequate resources. This action affects response time and energy consumption, often resulting in spikes. Service Level Agreement (SLA) violation refers to the number of destroyed containers that exceed their SLA deadline within an interval, as shown in Equation (5).

$$slaviolations_t = |\{c \in destroyed_t | c.destroyAt > SLA_t\}| \tag{5}$$

Where $c.destroyAt$ = number of containers destroyed at time interval t; SLA_t=acceptable SLA deadline at time t.

The average migration time refers to the average migration time of destroyed containers in an interval, as given in Equation (6).

$$avemigrationtime_t = \begin{cases} \frac{1}{|destroyed_t|} \sum_{c \in destroyed_t} T_c^{mig}, |destroyed_t| > 0 \\ 0, \qquad \text{otherwise} \end{cases} \tag{6}$$

Where C = containers destroyed; T_c^{mig} = migration time per container

3.3. Task Scheduling Decision

In the fog layer (Figure 2), an array of VMs handles tasks of maximum bound w with critical resources such as memory, IPS, bandwidth, and disk. Efficient scheduling is therefore required to achieve optimal QoS.

This is achieved using the dynamic programming algorithm (Section 3.2) with Equations (8), (9), and (10). In this research, the scheduler is denoted as S, such that $S: U_k \rightarrow U_k * VM$ holds when the allocation is possible based on the condition that the VM meets the resource requirement by the scheduled task. The scheduling problem is mathematically formulated by numbering the VMs from $j = 1, \dots, n$, and represented by a binary vector:

$$x_j = \begin{cases} 1, & \text{if } j-\text{th } VM \text{ is alocated} \\ 0, & \text{if } j-\text{th } VM \text{ is not allocated} \end{cases} \tag{7}$$

If the optimization (profit) of VM_j is p_j, its capacity c_j and the size of the task w, then the scheduler selects binary vectors x_j such that:

$$D_1 = \sum_{j=1}^{n} w_j x_j \leq c \tag{8}$$

While optimizing the QoS:

$$D_2 = \sum_{j=1}^{n} p_j x_j \leq c \tag{9}$$

Where D_2 is the value of the QoS parameter, such as response time or energy consumption [42].

3.5. Dynamic Programming Procedure

Given scalar values $m(1 \leq m \leq n)$, and $C(0 \leq C \leq c)$, let $fm(C)$ denote the optimal solution value:

$$f_m(C) = max\left\{\sum_{j=1}^{m} p_j x_j : \sum_{j=1}^{m} w_j x_j \leq C, x_j \in \{0,1\}, j = 1, \dots . m\right\} \tag{10}$$

This implies:

$$f1(C) = \begin{cases} 0, & \text{for } C = 0, \dots . , w_1 - 1 \\ p_1, & \text{for } C = w_1, \dots . , c \end{cases} \tag{11}$$

The dynamic programming in Figure 2 is constructed by splitting problems into n-stages for m ranging from 1 to n, and computing $fm(C)$ for $C = 0$; w at each stage.

$$fm(C) \begin{cases} f_m(C), & C = 0, \dots , w_m - 1 \\ max(f_{m-1}(C), f_{m-1}(C - w_m) + p_m), & C = w_m, \dots , c \end{cases} \tag{12}$$

The optimal state is computed as:

$$v = \min\left(\sum_{j=1}^{m} w_j, c\right) \tag{13}$$

Others parameters:

$$b = 2^{m-1} \tag{14}$$

$$p_C = f_{m-1}(C), \quad C = 0, \dots , v \tag{15}$$

$$x_C = \{x_{m-1}, x_{m-2}, \dots , x_1\}, \quad C = 0, \dots , v \tag{16}$$

Furthermore:

$$C = \sum_{j=1}^{m} w_j, x_j, \quad f_{m-1}(C) = \sum_{j-1}^{m-1} p_j, x_j \tag{17}$$

The dynamic programming algorithm is summarized in Algorithm 1.

Algorithm 1. Dynamic Programming

```
procedure DYNAMIC_PROGRAMMING
INPUT: n, c, p_j, w_j                 // j = 1..n
OUTPUT: z, x_j                        // optimal solution and binary assignment vector
 1: begin
 2:     // initialization for capacity 0..w1 − 1
 3:     for C := 0 to w_1 − 1 do
 4:         p_C := 0
 5:         x_C := 0
 6:     end for
 7:     // base case for capacity w1
 8:     v := w_1
 9:     b := 2
10:     p_v := p_1
11:     x_v := 1
12:     // iterate over items/VMs 2..n
13:     for m := 2 to n do
14:         call ITERATION1(v, b, p_C, x_C, w_m, p_m; v, b, p_C, x_C)
15:     end for
16:     z := p_c                      // objective value at capacity C = c
17:     determine x_j by decoding x_c
18: end

procedure ITERATION1
INPUT: v, b, p_C, x_C, w_m, p_m
OUTPUT: v, b, p_C, x_C
19: begin
20:     // expand active capacity range up to min(v + w_m, c)
21:     if v < c then
22:         u := v
23:         v := min(v + w_m, c)
24:         for C := u + 1 to v do
25:             p_C := p_u
26:             x_C := x_u
27:         end for
28:     end if
29:     // knapsack relaxation (backward) to include item m
30:     for C := v downto w_m do
31:         if p_C < p_{C − w_m} + p_m then
32:             p_C := p_{C − w_m} + p_m
33:             x_C := x_{C − w_m} + b
34:         end if
35:     end for
36:     b := 2 * b
37: end
```

The dynamic programming algorithm is iteratively executed over the tasks and hosts (Algorithm 1, line 15) to generate the decision sets $D^k = \{v, p_c\}$ These decision sets are

then provided as input to the LSTM model (Figure 2) to predict the QoS value $O(P_k)$ defined in Equation (26). The goal is to minimize the target objective function, which corresponds to either response time or energy consumption, as illustrated in Equation (27). This study is the first to integrate dynamic programming with LSTM to optimize IoT response time and energy consumption. Furthermore, it is also the first to simulate hepatitis sensors using JSON, thereby demonstrating how real-life hepatitis sensors can be created to support remote patient monitoring.

3.6. Long Short-Term Memory (LSTM) Model

The key concept of LSTM is the use of memory cells and gates that decide which information should be stored at any time [43]. Let $x_t, h_t,$ and y_t denote the input feature vector, hidden state, and output vector, respectively. Given an input sequence $(x_1, x_2, \dots, x_T)$, the corresponding hidden states and outputs are computed by iterating (18) and (19) for $t = 1, \dots, T$:

$$h_t = f_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{18}$$

$$y_t = W_{hy}h_t + b_y \tag{19}$$

Where W_{xh} is the input weight matrix; W_{hh} the recurrent weight matrix, and W_{hy} the output weight matrix; b_h and b_y are bias terms; and $f_h(\cdot)$ is an activation function. The LSTM implements the gate activations as:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \tag{20}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \tag{21}$$

$$\tilde{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \tag{22}$$

$$c_t = i_t \odot \tilde{c}_t + f_t \odot c_{t-1}, \tag{23}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \tag{24}$$

$$h_t = o_t \odot \tanh(c_t) \tag{25}$$

Where $\odot$ denotes element-wise multiplication; $W_{x\backslash *}$, $W_{h\backslash *}$ are weight matrices; $b_{\backslash *}$ are bias vectors; and i_t, f_t, $\tilde{c}_t$, c_t, o_t, h_t are the input gate, forget gate, candidate cell, memory cell, output gate, and hidden state, respectively. With h_t from (25), the output is obtained. In a single LSTM layer, a total of $4(NM + N^2 + N)$ parameters are learned (for input size M and hidden size N)

In this research, a finite number of task and host decision pairs from (10) (with a maximum limit M) are fed into the LSTM as D^k. At any interval T_k, the active tasks $|W_k| \leq M$. The utilization matrix of IPS, RAM, bandwidth, and disk for the active tasks forms a potential vector of size Z. Hence, the decision pairs produced by Dynamic Programming are encoded as the task-utilization matrix $\Phi(W_{k-1})$ of size $M \times Z$. Another vector encodes the virtual-machine utilization matrix (IPS, RAM, bandwidth, disk, capacities, and per-VM latency) of size Z' per VM. For $|VM|$ virtual machines, this yields a matrix $Z' \times |VM|$, denoted $\Phi(W_{k-1})$. Together with the decision matrix $\Phi(D)$, the LSTM input at interval T_k is the set of new tasks N_k, waiting tasks Q_k, active tasks W_k, completed tasks C_k, and $\Phi(D)$, The model is fed with $x = \Phi(W_{k-1}), \Phi(VM_{k-1}), \Phi(D)$ to estimate the QoS value $O(P_k)$ (latency or energy) as defined in (26) and illustrated in Figure 2.

The scenario is modeled as a continuous function $f(x; \theta)$ where the LSTM approximates $O(P_k)$ with θ as the learnable parameter vector and x the discrete/continuous input

within a defined domain. Specifically, x comprises the utilization matrices of tasks and VMs together with the decision matrix. The parameter θ is learned using the dataset in (26):

$$x = \{LSTM[\Phi(W_{k-1}), \Phi(VM_{k-1}), \Phi(D)], O(P_k)\}_k \quad (26)$$

This dataset is generated by a simulated scheduler whose tasks are formulated using real traces from the Bitbrains dataset [44]. After training, θ enables the model to approximate a generic function over a large parameter set. The optimization problem is then to find the decision matrix $\Phi(D)$ that minimizes $f(x;\theta)$ as in (27):

$$\Phi(D)\text{minimize} \rightarrow f(x{:}\,\theta), \forall D \text{ satisfying (10) \& (16)} \quad (27)$$

3.7. Mean Squared Error (MSE) Loss Function

The LSTM used in this study is pre-trained; therefore, its predictive accuracy is evaluated based on the loss values generated during execution. The loss quantifies the discrepancy between predicted and ground-truth outputs. During prediction of the optimal value $O(P_k)$ (either response time or energy consumption) when the LSTM processes the dataset in (26), the loss is tracked using the MSE:

$$Loss(f(x{:}\,\theta), y) = \frac{1}{T}\sum_{t=0}^{T}\left(y - f(x{:}\,\theta)\right)^2 \quad (28)$$

Where T is the time interval, θ the LSTM parameter, and (x, y) are samples drawn from the dataset defined in (26).

3.8. Software and Hardware Used

All experiments were performed on the Python-based COSCO platform. COSCO provides a secure, agile, and scalable environment for building cloud-based IoT solutions in both simulated and real deployments. It also supports designing/programming sensors using Node.js via a JSON API. In this study, the sensors were configured to communicate with the cloud backend—specifically InfluxDB and Oracle-VM VirtualBox—to emulate realistic cloud–fog operations.

4. Results and Discussion

In cloud-based IoT environments, efficient resource management is essential for maintaining optimal performance. This study evaluates seven optimization algorithms, with a particular emphasis on response time and energy consumption, while also considering supporting indicators such as migration time, number of destroyed containers, and SLA violations. For benchmarking, well-known methods including GA, MILP, POND, GOBI, DRL, and GOBI2 are compared against the proposed Hybrid DP+LSTM. Table 1 summarizes these comparative results across all key performance indicators.

Table 1. Comparison of Algorithms Performance Metrics Evaluation for IoT Cloud.

Ref	Algorithm	Avg. Response Time (ms)	% Improvement Over DP/LSTM	Avg. Migration Time (ms)	Num. Destroyed	SLA Violations	Energy per Container (joules)
Ours	Hybrid DP+LSTM	82.91	100.0%	0.018	3	0	2,835,048
[9]	GOBI2	395.13	79.0%	0.083	2	0	3,664,337
[9]	GOBI	776.09	89.3%	0.849	3	0	4,463,010
[39]	DRL	512.41	83.8%	0.315	1	0	2,973,238
[32]	POND	2789.09	97.0%	2.653	6	2	3,310,966
[14]	MILP	35,720.49	99.8%	105.43	351	307	3,005,498
[35]	GA	145,465.53	99.9%	638.63	449	223	3,959,935

The discussion highlights how the hybrid approach not only reduces latency and energy use but also achieves more consistent performance across varying workloads. This comprehensive evaluation demonstrates that the integration of analytical and learning-based methods provides a balanced trade-off between efficiency and reliability in IoT healthcare systems.

4.1. Energy Interval Dissipated by Algorithms

Figure 3 represents the total energy interval dissipated by the respective algorithms. Most algorithms have similar energy consumption, except for Hybrid DP + LSTM, which is significantly lower, making it more energy-efficient.

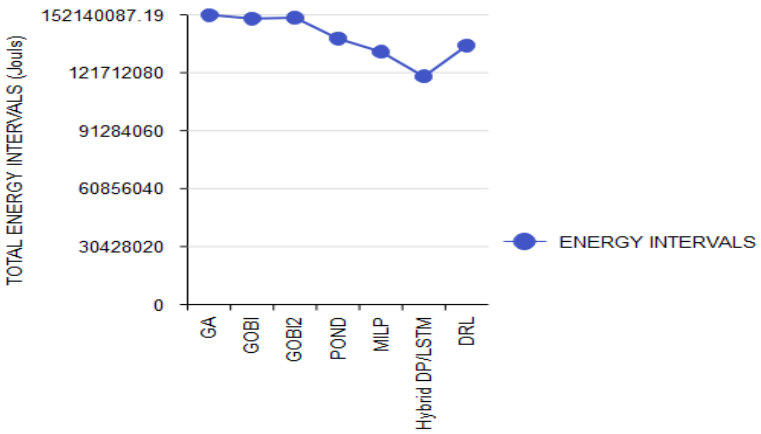

Figure 3. Total Energy Interval

In Figure 3, GA, GOBI, and GOBI2 show the highest energy consumption, indicating lower energy efficiency, which can negatively affect scalability and sustainability in fog environments. POND and MILP show noticeable energy reductions compared to the previous three. Hybrid DP + LSTM exhibits the lowest energy consumption, indicating the highest QoS efficiency in terms of energy, and making it highly suitable for energy-constrained fog nodes. DRL exhibits a slight increase in energy usage compared to Hybrid DP + LSTM, yet it remains more efficient than earlier algorithms, indicating a reasonable balance between energy use and other QoS metrics, such as adaptability and learning capability.

4.2. Average Response-Time Generated by Algorithms

Figure 4 depicts the average response time. Apparently, GA has the highest response time, indicating delays, while Hybrid DP + LSTM achieves the lowest response time, suggesting better efficiency. Table 2 presents the ratings of the algorithms' performance in a fog environment, with respect to their respective response times.

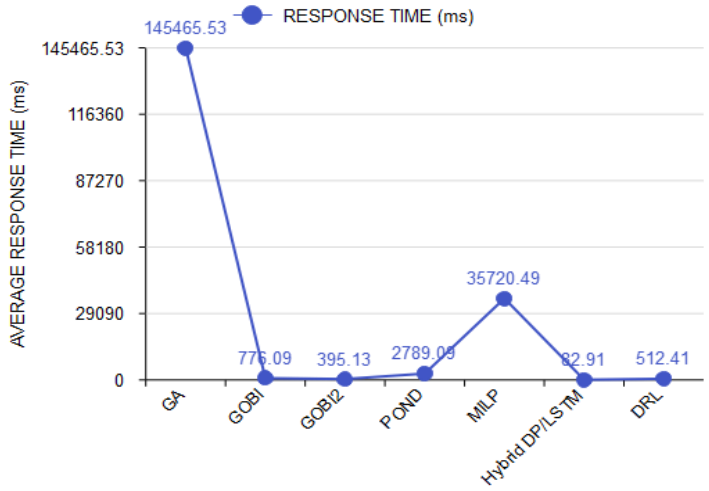

Figure 4. Average Response Time

Table 2. Algorithm comparison based on response time.

Algorithm	Avg. Response Time (ms)	QoS Evaluation	Observation
Hybrid DP + LSTM	82.91	Excellent (Real-time)	Best latency; ideal for time-sensitive applications
GOBI2	395.13	Very Good	Optimized version of GOBI; low latency
GOBI	776.09	Good	Decent latency, may lack adaptability
DRL	512.41	Very Good	Adaptive, intelligent decision-making
POND	2,789.09	Moderate	Slower, possibly due to static scheduling
MILP	35,720.49	Poor	Computationally heavy; not scalable
GA	145,465.53	Very Poor	Extremely high latency; unsuitable for real-time use

4.3. Ablation of Hybrid DP +LSTM

Furthermore, an ablation of the Hybrid DP + LSTM was performed to obtain the results of the performances of DP-only and LSTM-only to validate if the performance of the Hybrid DP+LSTM to form a hybrid model is better than the performances of the individual algorithms, as shown in Table 3

Table 3. Ablation study results.

Components	Avg. Response Time (ms)	% Improvement Over DP/LSTM	Avg. Migration Time (ms)	Num. Destroyed	Energy per Container (joules)
Hybrid DP+LSTM	82.91	100%	0.0176	3	2,835,048.21
DP Only	156.57	47.05%	0.1943	1	2,640,936.98
LSTM Only	282.41	70.64%	0.0730	1	3,671,515.11

The results in Table 3 show that the hybrid DP+LSTM model achieves a 47.05% (156.57ms) improvement over DP and a 70.64% (282.41ms) improvement over LSTM in average response time, as it combines the deterministic optimization strength of Dynamic Programming with the predictive capabilities of LSTM. DP-only excels at finding near-optimal scheduling decisions through recursive cost minimization; however, it can be computationally intensive for large and dynamic fog workloads. LSTM, on the other hand, can learn temporal workload patterns and predict future task arrivals; however, it suffers from higher latency due to its less precise optimization. The hybridization enables DP to benefit from LSTM's foresight, allowing for faster and more adaptive placement decisions, which significantly lowers response time.

On the energy efficiency side, Hybrid DP+LSTM shows a 22.80% (3,671.51 joules) improvement over LSTM, but a 7.34% (2,640.93) underperformance compared to DP. This outcome reflects a trade-off: DP, by design, minimizes resource wastage through exact optimization, which explains its slightly lower energy consumption. LSTM introduces prediction-driven migrations that may increase overhead. When combined with DP, some of this predictive overhead remains, resulting in slightly higher energy usage compared to DP-only. However, the hybrid still avoids the excessive migrations and energy drain observed in LSTM-only scheduling, keeping energy efficiency within an acceptable range.

4.4. Hybrid DP+LSTM Model Accuracy

The LSTM is a pre-trained model that was combined with the DP. Hence, its knowledge was transferred to augment the task scheduling capacity of the DP, thereby improving the QoS of the IoT Cloud System. The accuracy of the model was estimated based on the Loss values generated and returned by the MSE function. The loss function value reflects the difference between the actual values and the predicted values by the LSTM. Therefore, higher loss spikes suggest poor prediction, and stable loss suggests accurate predictions. In Figure 5, the pre-trained state of the model is the key reason why loss started off low, dropped slightly in the first epoch, and then remained stable for the remaining epochs.

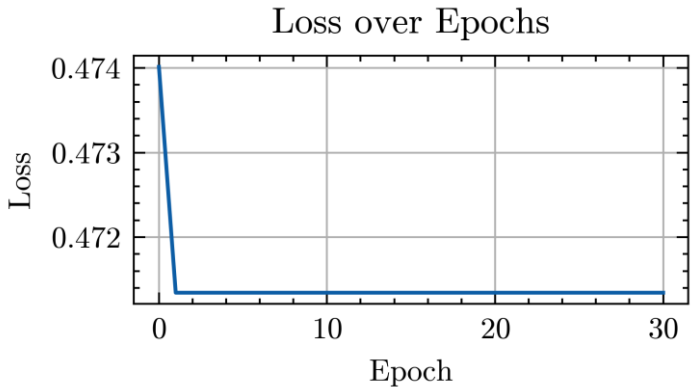

Figure 5. DP/LSTM Loss over Epoch

4.5. Analysis of Findings

From the analysis, Hybrid DP+LSTM demonstrated superior performance across key metrics, particularly in response time and migration time, which are critical for IoT Cloud environments. It achieved improvements of up to 99.94% and 99.8% over traditional methods, such as GA and MILP, respectively. Additionally, Hybrid DP+LSTM had no SLA violations and maintained a balanced energy consumption profile, making it a viable solution for large-scale IoT deployments. This makes it highly suitable for latency-sensitive applications such as telemedicine, real-time traffic management, and remote industrial monitoring.

- Adoption of Hybrid DP+LSTM in IoT Cloud: Given its superior performance, it is recommended for IoT Cloud infrastructures requiring low-latency response times, particularly in areas such as telemedicine, smart grid monitoring, and intelligent transportation systems.
- Further Optimization of GOBI2 and DRL: These algorithms performed relatively well but can be further optimized to improve response times for IoT services in critical applications, such as drone-based surveillance and automated robotic systems.
- Energy Optimization Strategies for IoT Devices: While Hybrid DP+LSTM was energy-efficient, further improvements in energy management are needed to enhance sustainability in IoT Cloud environments, especially in large-scale industrial IoT (IIoT) applications.
- Application-Specific Algorithm Selection: For IoT scenarios with minimal SLA concerns, POND and GOBI2 can be considered as alternatives for workload balancing in environments such as smart homes and connected agriculture.
- DP-only is best suited for pure energy minimization, but it is slower in response time.
- LSTM-only adapts to workload trends but suffers from high latency and energy inefficiency.

These findings offer valuable insights into workload distribution algorithms, facilitating the selection of the most suitable method for optimizing IoT Cloud performance and ensuring reliable service delivery in latency-sensitive applications.

5. Conclusions

The Hybrid DP+LSTM algorithm emerged as the most efficient and balanced solution. It delivered the lowest response time (82.91ms), zero SLA violations, the least energy consumption per container, and zero migration time. These results are especially significant for real-time, energy-sensitive IoT applications such as smart healthcare, autonomous vehicles, smart cities, emergency response systems, precision agriculture monitoring and industrial automation. The model's integration of predictive LSTM capabilities with DP's decision optimization allows it to anticipate workload demands and allocate resources proactively, ensuring service continuity and minimizing latency. DRL also demonstrated robust performance, particularly in terms of adaptability and low wait/migration times, making it a suitable option in dynamic environments. Although it consumed slightly more energy than Hybrid DP+LSTM, its learning-based decision-making provided reliable QoS in volatile fog settings. Algorithms like GOBI2 and POND offered moderate trade-offs and demonstrated improvements over

their predecessors, particularly in terms of energy and migration efficiency. In contrast, traditional optimization methods, such as GA and MILP, have demonstrated significant limitations. GA exhibited the highest container destruction, migration time, and response delay, indicating inefficiency and instability. MILP, although theoretically optimal, was too computationally intensive for time-sensitive deployments, resulting in the most SLA violations and increased wait times. These characteristics make both GA and MILP unsuitable for modern fog environments without extensive tuning or hybridization. In general, intelligent, learning-based algorithms, especially hybrid DP+LSTM and DRL are better suited for the distributed, real-time demands of fog computing. Their ability to adapt, predict, and make fast decisions ensures optimal QoS and sustainable resource use. The research proves that DP-only is best for pure energy minimization, but it is slower in response time. In contrast, LSTM adapts to workload trends but suffers from high latency and energy inefficiency. Hybrid DP+LSTM strikes a balance between the two, offering the lowest response time with moderate energy trade-offs, making it more suitable for fog environments where latency is critical. The future of fog and edge computing lies in such hybrid and AI-driven models that combine prediction, planning, and responsiveness to handle complex IoT workloads effectively.

Author Contributions: Conceptualization: NIS. and AEE.; Methodology: NIS and MEI.; Software: NIS.; Validation: MEI., AEE. and NIS.; Formal analysis: MEI and NIS.; Investigation: AEE.; Resources: NIS.; Data curation: NIS; Writing—original draft preparation: NIS.; Writing—review and editing: AEE and MEI.; Visualization: NIS.; Supervision: AEE.; Project administration: MEI. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used in this research are accessible from http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains.

Conflicts of Interest: The authors declare no conflict of interest in reporting the outcome of this research.

References

[1] W. A. Cruz Castañeda and P. Bertemes Filho, "Improvement of an Edge-IoT Architecture Driven by Artificial Intelligence for Smart-Health Chronic Disease Management," *Sensors*, vol. 24, no. 24, p. 7965, Dec. 2024, doi: 10.3390/s24247965.

[2] I. Batool, "RealTime Health Monitoring Using 5G Networks: A Deep Learning-Based Architecture for Remote Patient Care," *arXiv*, Jan. 2025, [Online]. Available: http://arxiv.org/abs/2501.01027

[3] R.-H. Hwang, Y.-C. Lai, and Y.-D. Lin, "Offloading Optimization with Delay Constraint in the 3-tier Federated Cloud, Edge, and Fog Systems," in *2021 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2021, pp. 1–6. doi: 10.1109/GLOBECOM46510.2021.9685111.

[4] N. Gholipour, M. D. de Assuncao, P. Agarwal, J. Gascon-Samson, and R. Buyya, "TPTO: A Transformer-PPO based Task Offloading Solution for Edge Computing Environments," in *2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS)*, Dec. 2023, pp. 1115–1122. doi: 10.1109/ICPADS60453.2023.00164.

[5] U. K. Lilhore *et al.*, "Cloud-edge hybrid deep learning framework for scalable IoT resource optimization," *J. Cloud Comput.*, vol. 14, no. 1, p. 5, Feb. 2025, doi: 10.1186/s13677-025-00729-w.

[6] A. Cotorobai, J. M. Silva, and J. L. Oliveira, "A Federated Random Forest Solution for Secure Distributed Machine Learning," in *2025 IEEE 38th International Symposium on Computer-Based Medical Systems (CBMS)*, Jun. 2025, pp. 769–774. doi: 10.1109/CBMS65348.2025.00159.

[7] A. Hennebelle, Q. Dieng, L. Ismail, and R. Buyya, "SmartEdge: Smart Healthcare End-to-End Integrated Edge and Cloud Computing System for Diabetes Prediction Enabled by Ensemble Machine Learning," in *2024 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec. 2024, pp. 127–134. doi: 10.1109/CloudCom62794.2024.00031.

[8] T. E. Ali, F. I. Ali, P. Dakić, and A. D. Zoltan, "Trends, prospects, challenges, and security in the healthcare internet of things," *Computing*, vol. 107, no. 1, p. 28, Jan. 2025, doi: 10.1007/s00607-024-01352-4.

[9] S. Tuli, S. R. Poojara, S. N. Srirama, G. Casale, and N. R. Jennings, "COSCO: Container Orchestration Using Co-Simulation and Gradient Based Optimization for Fog Computing Environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 1, pp. 101–116, Jan. 2022, doi: 10.1109/TPDS.2021.3087349.

[10] Y. Gu *et al.*, "Deep Reinforcement Learning for Job Scheduling and Resource Management in Cloud Computing: An Algorithm-Level Review," *arXiv*. Jan. 02, 2025. [Online]. Available: http://arxiv.org/abs/2501.01007

[11] J. V Guttag, *Introduction to Computation and Programming Using Python: With Application to Computational Modeling and Understanding Data*, 3rd ed. The MIT Press, 2021.

[12] A. S, A. Geetha, R. K, S. S, and S. D, "Latency Reduction in Medical IoT Using Fuzzy Systems by Enabling Optimized Fog Computing," *Int. J. Electr. Electron. Eng.*, vol. 9, no. 12, pp. 156–166, Dec. 2022, doi: 10.14445/23488379/IJEEE-V9I12P114.
[13] M. Kumar *et al.*, "Healthcare Internet of Things (H-IoT): Current Trends, Future Prospects, Applications, Challenges, and Security Issues," *Electronics*, vol. 12, no. 9, p. 2050, Apr. 2023, doi: 10.3390/electronics12092050.
[14] H. A. Alharbi, B. A. Yosuf, M. Aldossary, and J. Almutairi, "Energy and Latency Optimization in Edge-Fog-Cloud Computing for the Internet of Medical Things," *Comput. Syst. Sci. Eng.*, vol. 47, no. 1, pp. 1299–1319, 2023, doi: 10.32604/csse.2023.039367.
[15] N. Mishra and S. Pandya, "Internet of things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review," *IEEE Access*, vol. 9, pp. 59353–59377, Jul. 2021, doi: 10.1109/ACCESS.2021.3073408.
[16] M. M. Islam, S. Nooruddin, F. Karray, and G. Muhammad, "Internet of things: Device capabilities, architectures, protocols, and smart applications in healthcare domain," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3611–3641, Jan. 2022, doi: 10.1109/JIOT.2022.3228795.
[17] B. Pradhan, S. Bhattacharyya, and K. Pal, "IoT-Based Applications in Healthcare Devices," *J. Healthc. Eng.*, vol. 2021, pp. 1–18, Mar. 2021, doi: 10.1155/2021/6632599.
[18] K. Bagneid, Y. Sherif, M. Soliman, and M. Hussein, "Design, Simulation, and Implementation of Connected IoT Wearable Devices in Healthcare Applications," *Proc. Pakistan Acad. Sci. A. Phys. Comput. Sci.*, vol. 58, no. 3, pp. 59–65, Feb. 2022, doi: 10.53560/PPASA(58-3)745.
[19] M. Aldossary, "Multi-Layer Fog-Cloud Architecture for Optimizing the Placement of IoT Applications in Smart Cities," *Comput. Mater. Contin.*, vol. 75, no. 1, pp. 633–649, 2023, doi: 10.32604/cmc.2023.035414.
[20] Y. Wang, S. Wang, B. Yang, B. Gao, and S. Wang, "An effective adaptive adjustment method for service composition exception handling in cloud manufacturing," *J. Intell. Manuf.*, vol. 33, no. 3, pp. 735–751, Mar. 2022, doi: 10.1007/s10845-020-01652-4.
[21] E. Yaacoub, K. Abualsaud, T. Khattab, and A. Chehab, "Secure Transmission of IoT mHealth Patient Monitoring Data from Remote Areas Using DTN," *IEEE Netw.*, vol. 34, no. 5, pp. 226–231, Sep. 2020, doi: 10.1109/MNET.011.1900627.
[22] A. Mukherjee, S. Ghosh, A. Behere, S. K. Ghosh, and R. Buyya, "Internet of Health Things (IoHT) for personalized health care using integrated edge-fog-cloud network," *J. Ambient Intell. Humaniz. Comput.*, vol. 12, no. 1, pp. 943–959, Jan. 2021, doi: 10.1007/s12652-020-02113-9.
[23] A. Cook *et al.*, "Internet of Cloud: Security and Privacy Issues," in *Cloud Computing for Optimization: Foundations, Applications, and Challenges*, 2018, pp. 271–301. doi: 10.1007/978-3-319-73676-1_11.
[24] J. L. Shah, H. F. Bhat, and A. I. Khan, "Integration of Cloud and IoT for smart e-healthcare," in *Healthcare Paradigms in the Internet of Things Ecosystem*, Elsevier, 2021, pp. 101–136. doi: 10.1016/B978-0-12-819664-9.00006-5.
[25] H. Y. Y. Nyein *et al.*, "A wearable patch for continuous analysis of thermoregulatory sweat at rest," *Nat. Commun.*, vol. 12, no. 1, p. 1823, Mar. 2021, doi: 10.1038/s41467-021-22109-z.
[26] S. Rahman, M. Irfan, M. Raza, K. Moyeezullah Ghori, S. Yaqoob, and M. Awais, "Performance Analysis of Boosting Classifiers in Recognizing Activities of Daily Living," *Int. J. Environ. Res. Public Health*, vol. 17, no. 3, p. 1082, Feb. 2020, doi: 10.3390/ijerph17031082.
[27] A. Zollanvari, *Machine Learning with Python: Theory and Implementation*. Cham: Springer International Publishing, 2023. doi: 10.1007/978-3-031-33342-2.
[28] N. Singh, M. Raza, V. V. Paranthaman, M. Awais, M. Khalid, and E. Javed, "Internet of Things and cloud computing," in *Digital Health*, Elsevier, 2021, pp. 151–162. doi: 10.1016/B978-0-12-818914-6.00013-2.
[29] D. Bălăcian and S. Stancu, "A Performance-Driven Economic Analysis of a LSTM Neural Network Used for Predicting Building Energy Consumption," *Proc. Int. Conf. Bus. Excell.*, vol. 17, no. 1, pp. 29–37, Jul. 2023, doi: 10.2478/picbe-2023-0005.
[30] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic Scheduling for Stochastic Edge-Cloud Computing Environments Using A3C Learning and Residual Recurrent Neural Networks," *IEEE Trans. Mob. Comput.*, vol. 21, no. 3, pp. 940–954, Mar. 2022, doi: 10.1109/TMC.2020.3017079.
[31] S. S. Gill *et al.*, "Transformative effects of IoT, Blockchain and Artificial Intelligence on cloud computing: Evolution, vision, trends and open challenges," *Internet of Things*, vol. 8, p. 100118, Dec. 2019, doi: 10.1016/j.iot.2019.100118.
[32] X. Liu, B. Li, P. Shi, and L. Ying, "POND: Pessimistic-Optimistic oNline Dispatching," *arXiv*. May 11, 2021. [Online]. Available: http://arxiv.org/abs/2010.09995
[33] G. K. Shrivastava, P. Kaushik, and R. K. Pateriya, "Comprehensive Analysis of Web Page Classifier for Fsocused Crawler," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 9, pp. 57–65, Jul. 2019, doi: 10.35940/ijitee.I7477.078919.
[34] S. Bosmans, S. Mercelis, J. Denil, and P. Hellinckx, "Testing IoT systems using a hybrid simulation based testing approach," *Computing*, vol. 101, no. 7, pp. 857–872, Jul. 2019, doi: 10.1007/s00607-018-0650-5.
[35] S. Singh, A. S. Nandan, G. Sikka, A. Malik, and A. Vidyarthi, "A secure energy-efficient routing protocol for disease data transmission using IoMT," *Comput. Electr. Eng.*, vol. 101, p. 108113, Jul. 2022, doi: 10.1016/j.compeleceng.2022.108113.
[36] I. Ullah, N. U. Amin, M. A. Khan, H. Khattak, and S. Kumari, "An Efficient and Provable Secure Certificate-Based Combined Signature, Encryption and Signcryption Scheme for Internet of Things (IoT) in Mobile Health (M-Health) System," *J. Med. Syst.*, vol. 45, no. 1, p. 4, Jan. 2021, doi: 10.1007/s10916-020-01658-8.
[37] C. Mangla, S. Rani, and N. Herencsar, "An energy-efficient and secure framework for IoMT: An application of smart cities," *Sustain. Energy Technol. Assessments*, vol. 53, p. 102335, Oct. 2022, doi: 10.1016/j.seta.2022.102335.
[38] N. Singh and A. K. Das, "Energy-efficient fuzzy data offloading for IoMT," *Comput. Networks*, vol. 213, p. 109127, Aug. 2022, doi: 10.1016/j.comnet.2022.109127.
[39] H. Zhou, Z. Zhang, Y. Wu, M. Dong, and V. C. M. Leung, "Energy Efficient Joint Computation Offloading and Service Caching for Mobile Edge Computing: A Deep Reinforcement Learning Approach," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 950–961, Jun. 2023, doi: 10.1109/TGCN.2022.3186403.

[40] K. Alatoun, K. Matrouk, M. A. Mohammed, J. Nedoma, R. Martinek, and P. Zmij, "A Novel Low-Latency and Energy-Efficient Task Scheduling Framework for Internet of Medical Things in an Edge Fog Cloud System," *Sensors*, vol. 22, no. 14, p. 5327, Jul. 2022, doi: 10.3390/s22145327.

[41] H. U. Atiq, Z. Ahmad, S. K. uz Zaman, M. A. Khan, A. A. Shaikh, and A. Al-Rasheed, "Reliable Resource Allocation and Management for IoT Transportation Using Fog Computing," *Electronics*, vol. 12, no. 6, p. 1452, Mar. 2023, doi: 10.3390/electronics12061452.

[42] S. Aiswarya, K. Ramesh, and S. Sasikumar S, "IoT based Big data Analytics in Healthcare: A Survey," in *Proceedings of the Fist International Conference on Advanced Scientific Innovation in Science, Engineering and Technology, ICASISET 2020, 16-17 May 2020, Chennai, India*, 2021. doi: 10.4108/eai.16-5-2020.2304020.

[43] B. Sirisha, K. K. C. Goud, and B. T. . S. Rohit, "A Deep Stacked Bidirectional LSTM (SBiLSTM) Model for Petroleum Production Forecasting," *Procedia Comput. Sci.*, vol. 218, pp. 2767–2775, 2023, doi: 10.1016/j.procs.2023.01.248.

[44] D. Kshatriya and V. A. Lepakshi, "An Efficient Hybrid Scheduling Framework for Optimal Workload Execution in Federated Clouds to Maintain Performance SLAs," *J. Grid Comput.*, vol. 21, no. 3, p. 47, Sep. 2023, doi: 10.1007/s10723-023-09682-x.