Journal of
Sensor and Actuator Networks

MDPI

Article

Service-Aware Hierarchical Fog–Cloud Resource Mapping for e-Health with Enhanced-Kernel SVM

Alaa AlZailaa [1], Hao Ran Chi [2], Ayman Radwan [3,*] and Rui L. Aguiar [1]

[1] Instituto de Telecomunicações and DETI, Universidade de Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal; alz@ua.pt (A.A.); ruilaa@ua.pt (R.L.A.)
[2] Instituto de Telecomunicações and Universidade de Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal; ytchr@av.it.pt
[3] Department of Electrical and Computer Engineering and Instituto de Telecomunicações, Instituto Superior Técnico, University of Lisbon, 1049-001 Lisbon, Portugal
* Correspondence: aradwan@av.it.pt

Abstract: Fog–cloud-based hierarchical task-scheduling methods are embracing significant challenges to support e-Health applications due to the large number of users, high task diversity, and harsher service-level requirements. Addressing the challenges of fog–cloud integration, this paper proposes a new service/network-aware fog–cloud hierarchical resource-mapping scheme, which achieves optimized resource utilization efficiency and minimized latency for service-level critical tasks in e-Health applications. Concretely, we develop a service/network-aware task classification algorithm. We adopt support vector machine as a backbone with fast computational speed to support real-time task scheduling, and we develop a new kernel, fusing convolution, cross-correlation, and auto-correlation, to gain enhanced specificity and sensitivity. Based on task classification, we propose task priority assignment and resource-mapping algorithms, which aim to achieve minimized overall latency for critical tasks and improve resource utilization efficiency. Simulation results showcase that the proposed algorithm is able to achieve average execution times for critical/non-critical tasks of 0.23/0.50 ms in diverse networking setups, which surpass the benchmark scheme by 73.88%/52.01%, respectively.

Keywords: fog–cloud computing; task scheduling; service aware; support vector machine; network optimization; e-Health

check for updates

Citation: AlZailaa, A.; Chi, H.R.; Radwan, A.; Aguiar, R.L. Service-Aware Hierarchical Fog–Cloud Resource Mapping for e-Health with Enhanced-Kernel SVM. *J. Sens. Actuator Netw.* **2024**, *13*, 10. https://doi.org/10.3390/jsan13010010

Academic Editors: Stefan Fischer, Yichuang Sun, Haeyoung Lee and Oluyomi Simpson

Received: 7 October 2023
Revised: 14 January 2024
Accepted: 18 January 2024
Published: 1 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

e-Health is becoming the backbone for upgrading the conventional healthcare system by validating remote services, which are specifically important under pandemic conditions (e.g., COVID-19) to release the pressure of lack of trained healthcare professionals [1,2].

Predicted to process 92% of the overall workload within 5 years, cloud computing has proved its high centralized computational and storage capacity in e-Health, e.g., developing health information technology (HIT) systems [3]. Additionally, to prevent inefficient massive data aggregation to centralized cloud computing in the conceived ultra-large-scale e-Health system with the rapid increase in the number of users, fog computing has emerged, since it is located physically closer to users [4]. To integrate the aforementioned strengths of cloud and fog computing, the fog–cloud hierarchical structure, as well as efficient fog–cloud resource management, has provided a glimmer of hope to support high portability and automatic provisioning for future e-Health development [5]. There are plenty of efforts for resource management among fog and cloud nodes, e.g., resource mapping [6] and task scheduling [4], where task classification is essential to identifying the features of both computing nodes (e.g., computational capacity, potential latency, etc.) [7] and tasks (e.g., payload, etc.) [8].

Despite the foreseen advantages, the fog–cloud hierarchical structure for e-Health is a significantly complicated system, as a cross-layer optimization problem, with new challenges. To start, the cross-layer task scheduling of the fog–cloud hierarchical structure for e-Health gains an irregular solution space due to the quantitatively large difference between features of fog and cloud nodes. Therefore, it requires higher complexity of modeling and solving, which might fail to support critical/latency-sensitive tasks due to tedious computation. To the best of the authors' knowledge, there is still a lack of research efforts for the real-time task scheduling of fog–cloud hierarchical structures for e-Health services. Additionally, task classification also encounters serious computational complexity, particularly for the foreseen ultra-large-scale e-Health systems. Improper task classification directly results in inefficient resource utilization regarding diverse demands for task scheduling, optimization of QoS, and latency. Moreover, due to the uniqueness of e-Health services, critical tasks are defined with very low latency tolerance margin, which is determined by services/patients' information, e.g., medical records/real-time symptoms [9]. Therefore, task scheduling in e-Health should fuse features at the service level and network level simultaneously, which directly complicates the computational process.

Targeting the aforementioned challenges, we propose an e-Health infrastructure for real-time fog–cloud hierarchical task scheduling by dynamically considering the real-time requirements of tasks with the modeling of both networking and computation simultaneously. The contributions in the paper can be summarized as follows:

1. We propose a task classification algorithm, fusing features at the network level and service level for e-Health, which is efficient in achieving user-centric QoS maximization, with latency minimized for critical tasks. Support vector machine (SVM)-based task classification which is efficient in handling the defined latency-sensitive critical tasks is proposed. It is necessary to note that although deep learning algorithms increasingly gain markets, shallow machine learning (e.g., SVM) with low computational costs still presents strengths for latency-sensitive e-Health applications [5].
2. A new kernel type is proposed for comprehensively classifying network-level and service-level features, fusing convolution, cross-correlation, and auto-correlation, which gains high overall classification accuracy for specificity and sensitivity enhancement.
3. We propose a task priority assignment algorithm and a resource-mapping algorithm, which achieve sufficient overall latency for the defined critical tasks while improving the overall resource utilization efficiency.

The rest of the paper is organized as follows: Section 2 presents a survey of related efforts. Section 3 introduces the system modeling with the formulation of communication and computational processes. Section 4 proposes optimization modeling and the SVM-based task-scheduling algorithm, with a newly developed kernel type to optimize resource utilization efficiency and communication latency. Section 5 explores three key algorithms. These algorithms collaboratively prioritize tasks, classify resources, and effectively allocate tasks to suitable fog and cloud nodes. Section 6 showcases the result analysis of the proposed scheme. Section 7 concludes the paper.

2. Related Work

Task scheduling for fog–cloudcomputing is highly demanded due to uneven distribution of workload and heterogeneous computing capacity. Conventionally, task scheduling is performed among cloud nodes to improve computational efficiency. For instance, tasks are prioritized according to their payload, which is further scheduled based on the cloud's MIPS, ensuring processing reliability for tasks with high priority [10]. Compared with cloud computing, the popularity of fog computing increases demand for task scheduling, resulting from the limited computational capacity of individual fog nodes (FNs) when dealing with heterogeneous data densification for e-Health. Depending on criteria such as user-preference-oriented features of FNs [11], energy efficiency maximization [12], and overall latency minimization [13], fog-based task scheduling is modeled and solved. As illustrated in Table 1, critical network- and service-level fac-

tors that comprehensively cover the typical demands and indicators to achieve optimal service-aware fog–cloud resource mapping for e-Health are selected and considered in this paper. Concretely, FN capacity relates to the number of fog nodes available, influencing performance and task offloading decisions. Task features, encompassing computational complexity, data size, and latency requirements, are the essential criteria to be considered when achieving service-level task scheduling in the network layer. Task priority is a critical indicator to guide scheduling based on service-level urgency. Latency, i.e., delay in task completion, impacts the quality of service, particularly in e-Health scenarios. In addition, execution time is specifically considered, besides overall latency, to further highlight the impact of task complexity and fog node resources simultaneously. Network and computation modeling are responsible for ensuring accurate latency and execution time predictions in service-aware task scheduling, respectively. Offloading decisions to the cloud, in contrast to fog-level offloading, should be considered a key criterion in fog–cloud resource mapping, with the cloud providing centralized computational/storage power to data-intensive tasks for a trade-off among network latency, balancing proximity, and computational power.

Many current research efforts focused on either fog-based or cloud-based task scheduling, where nodes share similar features. For example, FNs normally gain a similar computational capacity level (i.e., much lower than cloud nodes), leading to a relatively regularly shaped optimization solution space for task classification [4]. For instance, a deep learning-based fog-computing architecture was proposed in order to diagnose heart diseases, which offloaded tasks based on the CPU load of FNs as a metric [14]. Similarly, greedy algorithm-based resource allocation and classification for FNs based on the availability of CPU and bandwidth were proposed [15].

The fog–cloud-based hierarchical structure has drawn significant research attention in recent 5G development and research on next-generation communications thanks to its benefits of enhanced connectivity between sensors/devices/users and computing nodes while reducing transmission and computational latency with high QoS. In fact, fog–cloud hierarchical task scheduling has been foreseen to be dominant for future heterogeneous network (HetNet) deployment, requiring systemic optimization regarding service-centric and user-centric performance, e.g., energy efficiency, QoS, overall latency, etc. [16]. However, complicated features with multi-layer modeling cannot be tackled with the above-mentioned homogeneousfog–cloud-only task scheduling. There have been attempts to construct hierarchies among cloud nodes (e.g., cloudlet [1]) or FNs (e.g., multi-layer FNs [17]), aiming to increase utilization efficiency while reducing latency, as these are still not replaceable in fog–cloud hierarchical task scheduling with more diverse features. The most adopted strategies for fog–cloud task scheduling generally follow the principle that latency-tolerant and large-size tasks are assigned to cloud nodes, and latency-sensitive tasks, to FNs [9], based on which minimizing the overall makespan, maximizing resource utilization efficiency, or load balancing is targeted [16,18–21].

To facilitate task scheduling in fog–cloud hierarchical e-Health systems, task classification is essential, with tasks being normally categorized/prioritized (e.g., critical, moderate, normal [18]). Concretely, the priority of tasks is mostly determined with reference to the overall latency requirement [19], deadline and available resources in FNs [22], payload [23], makespan constraints with available resources [20], or task length [24]. For instance, a mobility-aware scheduling scheme to dynamically distribute tasks to the fog or the cloud was proposed for e-Health [25] by prioritizing tasks based on data size, response time, and complexity. A static scheduling method in a fog–cloud heterogeneous environment was also proposed to reduce CPU execution time and network usage [26]; it classifies tasks according to the required MIPS and the trade-off between CPU execution time and allocation memory of FNs. Task offloading to cloud nodes is still based on the aforementioned "latency-tolerant and large-size tasks to cloud nodes" principle.

Table 1. Literature review.

Ref.	FN Capacity	Task Features	Task Priority	Latency	Execution Time	Network Modeling	Computation Modeling	Offloading to Cloud
[10]	✘	✔	✔	✘	✔	✘	✘	✘
[11]	✘	✘	✘	✔	✔	✘	✘	✘
[12]	✔	✔	✘	✔	✔	✘	✘	✘
[16]	✔	✘	✘	✔	✔	✔	✔	✘
[18]	✔	✔	✔	✘	✔	✘	✘	✔
[19]	✔	✘	✔	✘	✔	✔	✘	✔
[20]	✔	✘	✔	✔	✔	✔	✘	✔
[21]	✔	✘	✘	✔	✔	✔	✔	✘
[23]	✘	✔	✔	✔	✘	✘	✘	✘
Proposed	✔	✔	✔	✔	✔	✔	✔	✔

However, due to the nature of e-Health, besides network-level requirements, tasks should be further classified based on service-level demands, e.g., patients' profile, etc., which should play a vital role in minimizing critical tasks (e.g., time-sensitive tasks, emergencies, etc.). For instance, detection of abnormal cardiovascular conditions might be much more serious for patients with related EHR than for normal users, which cannot be reflected by network-level features. Some attempts were made to reduce the computational process for mobile patients based on offloading high-priority tasks to the core with higher computational power based on the predefined priority of tasks in the application layer [27]. Comprehensive determination of task priority is still missing for e-Health regarding both network-level requirements and service-level demands. Therefore, we propose a task-scheduling scheme, comprehensively considering network-level and service-level features for task classification.

3. Fog–Cloud Hierarchical Infrastructure and Modeling for e-Health

This section outlines an innovative task-scheduling framework in a fog–cloud hierarchy specifically designed to enhance e-Health services by optimizing task allocation and reducing latency.

3.1. Fog–Cloud Hierarchical Infrastructure for e-Health

We propose an optimal cross-layer task-scheduling scheme based on a generic fog–cloud hierarchical infrastructure for e-Health, as shown in Figure 1. e-Health devices in the IoT layer are connected to base stations (BSs), which further relay the aggregated data to the task orchestrator (also known as fog-layer broker), centralized for task scheduling. Allocation between e-Health devices and BSs can be achieved using our previously published schemes, achieving optimal link quality with interference mitigation [6]. Conventionally, task-scheduling methods either schedule tasks among fog nodes (FNs) or forward latency-insensitive tasks to cloud nodes, which results in inefficient resource utilization of the fog–cloud hierarchical infrastructure while potentially failing to support critical tasks in e-Health according to service-level demands. Therefore, we focus on cross-layer task scheduling by embedding a support vector machine (SVM)-based task classification algorithm at the orchestrator, the output of which guides real-time task scheduling to fog–cloud nodes. Compared with the conventional methods, the proposed algorithm is capable of achieving real-time fog–cloud cross-layer task scheduling with minimized overall latency for critical tasks in e-Health while ensuring efficient computation for all users. The proposed task classification and task-scheduling algorithm will be discussed in Sections 4 and 5, respectively.

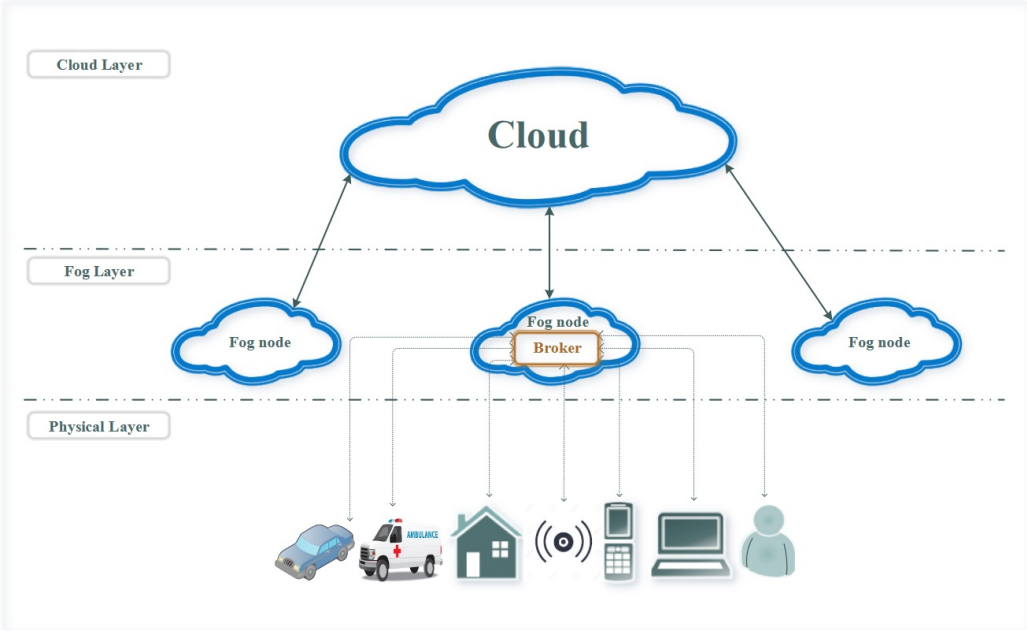

Figure 1. Generic cloud–fog hierarchical architecture for e-Health.

Suppose that there are M cloud nodes and N FNs in the infrastructure designed with heterogeneous computational capacities as

$$\begin{aligned} A_M^c &= \{a_1^c, a_2^c, \ldots, a_M^c\} \\ A_N^f &= \{a_1^f, a_2^f, \ldots, a_N^f\} \end{aligned} \tag{1}$$

where $M+N$ fog–cloud nodes support U_{IoT} e-Health devices, with each device u generating $x_1^u, x_2^u, \ldots, x_{K_u}^u$ tasks in unit time period t. To model this random generation, we have employed a Poisson distribution, which is commonly considered [28] and well proved to reflect the scenarios in our paper, i.e., events happening independently, at a steady rate, whose exact occurrence is random. Concretely, $D_{x_{k_u}^u}$ and $L_{x_{k_u}^u}$, with $k_u \leq K_u, u \leq U$, are the latency and the payload of the kth task generated by device u, respectively, which are well recognized for task classification, especially for emergent and critical e-Health applications [9]. However, specifically in e-Health, critical tasks should be defined according to service-level demands. For instance, it is intuitive that tasks related to acute diseases should gain higher critical level than chronic diseases, with much higher probability of leading to serious drawbacks.

To correlate tasks according to service-level demands, we consider the type of tasks according to their service-level functionality, $\boldsymbol{\theta}$, which holds four different values, defined as

$$\boldsymbol{\theta}_x^u = \{inquiry, backup, notification, alarm\} \tag{2}$$

where:

1. Inquiry: triggered for storing information in medical records;
2. Backup: generated periodically to update medical records;
3. Notification: set as reminders, e.g., pill time and therapy appointment for patients, medical status alert to medical workers, etc.;
4. Alarm: generated based on the diagnosis results, which are also affected/referenced by the proposed orchestrator, regarding task classification.

The final value of θ_x^u is set by the scheduler, after examining all indicated parameters in Algorithm 1.

Algorithm 1 Task priority determination algorithm.

/* 1.0 for each task $(x^u_{k_u})$ check $(\beta^u_{x_{k_u}})$, $(\mu^u_{x_{k_u}})$ fields */
if $(\beta^u_{x_{k_u}}) == (\mu_l^{\beta^u_x})$ **then**
    $SVM\_weight(x^u_{k_u}) == High$
else if $(\beta^u_{x_{k_u}})! = (\mu_l^{\beta^u_x})$ **then**
    $SVM\_weight(x^u_{k_u}) == Medium$
else
    $SVM\_weight(x^u_{k_u}) == Low$
end if
/* 2.0 assign the priority to the task */
$priority(x^u_{k_u}) = SVM_{weight}(x^u_k) + weight(PL(x^u_k))$
/* 2.1 label the task, assign the value (type of task) */
$\theta^u_{x_{k_u}} = \{inquiry, backup, notification, alarm\}$
/* 2.2 order tasks descending based on priority */
$P(x^u_k(p)) = \left\{x^u_k(p), x^u_{k-1}(p-1), \cdots, x^u_0(0)\right\}$
/* 3.0 send the prioritized tasks from the fog nodes */
$P(x^u_k(p)) \Rightarrow Orchestrator\{[X(p)^U_K][A^N_F]\}$

Additionally, patients' medical records, $\boldsymbol{\beta}^u_x$, are considered, in task scheduling, a key feature reflecting service-level demands and are generalized as acute and chronic diseases ($x \in \{x^u_1, \ldots, x^u_{K_u}\}, u \in \{1, \ldots, U_{IoT}\}$):

$$\boldsymbol{\beta}^u_x = \{Acute, Chronic\} \tag{3}$$

Similar to $\boldsymbol{\theta}$, tasks related to acute diseases should be considered critical tasks, compared with chronic diseases. To model $\boldsymbol{\beta}^u_x$ numerically, we define $\boldsymbol{\mu}$ as preliminary symptoms of diseases, according to patients' medical records:

$$\boldsymbol{\mu}^u = \begin{bmatrix} \mu_1^{\beta^u_1} & \cdots & \mu_L^{\beta^u_1} \\ \vdots & \ddots & \vdots \\ \mu_1^{\beta^u_{K_u}} & \cdots & \mu_L^{\beta^u_{K_u}} \end{bmatrix} \tag{4}$$

where $\forall u \in \{1, \ldots, U_{IoT}, \boldsymbol{\mu}^u \in \{0, 1\}$, with $\boldsymbol{\mu}^u = 1$ indicating that patient u has corresponding symptoms of acute or chronic diseases and $\boldsymbol{\mu}^u = 0$ indicating that no symptoms have been detected.

3.2. Network Modeling

Since $\boldsymbol{\theta}$ is considered a key parameter set for task scheduling, network modeling should be formulated to reflect the stringent latency and QoS requirements, which are modeled in a cross-layer fashion.

As illustrated in Figure 1, U_{IoT} devices are connected to FNs through a wireless network, with distance $\delta^f_{u,n}$ between device u and FN n. Similarly, with the conceived large throughput in the next generation of communications, the connection between fog and cloud layers is also assumed to be wireless, with distance being represented by $\delta^c_{n,m}$.

For each task $x^u_{k_u}, k_u \in \{1, \ldots, K_u\}, u \in \{1, \ldots, U_{IoT}\}$, the overall end-to-end transmission latency can be constrained as

$$D^u_{x_{k_u}} < \tau_{comm,x^u_{k_u}} + \tau_{wait,x^u_{k_u}} + \tau_{proc,x^u_{k_u}} \tag{5}$$

where $\tau_{comm,x^u_{k_u}}$, $\tau_{wait,x^u_{k_u}}$, and $\tau_{proc,x^u_{k_u}}$ refer to latency of propagation, waiting, and processing for task $x^u_{k_u}$, respectively.

$\tau_{comm,x_{k_u}^u}$ is affected by the transmission delay ($t_{trans,x_{k_u}^u}$), estimated as

$$t_{trans,x_{k_u}^u} = \frac{J_u^f}{R_n^{f,u}} + \frac{I_u^c}{R_m^{c,u}} \tag{6}$$

where $J_u^f = \sum x_{k_u}^u$ and $I_u^c = \sum x_{k_u}^u$ represent the number of bits of task $x_{k_u}^u$ from device u to FNs and cloud nodes, respectively. $R_n^{f,u}$ and $R_m^{c,u}$ refer to the data rates for transmitting the bits of $x_{k_u}^u$ to FN n and cloud nodes m, respectively, as [6]

$$R_n^{f,u} = (W_n^f) * \log(1 + SINR_n^{f,u}) \tag{7}$$

$$R_m^{c,u} = (W_m^c) * \log(1 + SINR_m^{c,u}) \tag{8}$$

where $W_i^z, i \in \{m,n\}, z \in \{c,f\}$, is the bandwidth assigned to node i and $SINR_i^{z,u}$ represents the signal-to-interference-plus-noise ratio for node i regarding device u. In the case of $i = n$, the $SINR$ is calculated as

$$SINR_n^{f,u} = \frac{Pw_u h_u (\delta_{u,n}^f)^{-\alpha_{pl}}}{\sigma^2 + I_n^f} \tag{9}$$

where Pw_u refers to the transmission power of device u (uplink considered). h_u is the channel power coefficient. Path loss is defined according to δ_u with path loss coefficient α_{pl}. σ^2 and I_n^f are the noise variance and the residual interference power for FN n, respectively [29]. Similarly, for the connection between device u and cloud node m, the SINR is calculated as

$$SINR_m^{c,u} = \frac{Pw_u h_u (\delta_{u,m}^c)^{-\alpha_{pl}}}{\sigma^2 + I_m^c} \tag{10}$$

Together with the propagation delay related to multi-hopping, $\phi_i^{z,u}, i \in \{m,n\}, z \in \{c,f\}$, has a proportional relationship with $\delta_{u,n}^f$ and $\delta_{n,m}^c$; $\tau_{comm,x_{k_u}^u}$ is derived as

$$\tau_{comm,x_{k_u}^u} = t_{trans,x_{k_u}^u} + \phi_i^{z,u} \quad i \in \{m,n\}, z \in \{c,f\} \tag{11}$$

$\tau_{wait,x_{k_u}^u}$ is derived as the summation of the delay for task $x_{k_u}^u$ requesting network access ($\gamma_{x_{k_u}^u}$) and the server response time ($\nu_{i,x_{k_u}^{z,u}}$) required for computing nodes to respond to the task:

$$\tau_{wait,x_{k_u}^u} = \gamma_{x_{k_u}^u} + \nu_{i,x_{k_u}^{z,u}} \quad i \in \{m,n\}, z \in \{c,f\} \tag{12}$$

$\tau_{proc,x_{k_u}^u}$ is affected by the payload of task $x_{k_u}^u$ and the computing capacity of the associated computing nodes, $i \in \{m,n\}, z \in \{c,f\}$, and is calculated as

$$\tau_{proc,x_{k_u}^u} = \frac{L_{x_{k_u}}^u}{a_i^z}, \quad i \in \{m,n\}, z \in \{c,f\} \tag{13}$$

3.3. Computation Modeling

Both cloud and fog nodes share basic features, e.g., computing capacity, including virtual central processing unit (vCPU) cores, MIPS, RAM, and storage, regarding A_n^f and A_m^c defined in (1):

$$A_i^z = \sum \{vCPU + MIPS + RAM + storage\}, i \in \{m,n\}, z \in \{c,f\} \tag{14}$$

We suppose that $T^z_{u,i}$, $i \in \{m,n\}, z \in \{c,f\}$, refers to the total number of time slots in the computing node, where

$$T^z_{u,i} = \sum_{t=1}^{T} \Delta\tau^z_{u,i}(t), \quad Z = c, f. \tag{15}$$

where T is the total processing time. $\Delta\tau^z_{u,i}(t)$ is the available time of computing node i in time slot t.

The required computing resource for task $x^u_{k_u}$ is defined as $rq^u_{x^u_{k_u}}(t)$ in a given time slot t. It is highlighted that $rq^u_{x^u_{k_u}}(t)$ is heterogeneous for diverse devices, which complicates task scheduling, with user/service-centric QoS and latency optimization. $rq^u_{x^u_{k_u}}(t)$ is constrained as

$$\forall u \in \{1, \ldots, U_{IoT}\} : \sum_{k=1}^{K_u} \sum_{t=1}^{T} rq^u_{x^u_{k_u}}(t) \leq \sum_{i=1}^{N} a^f_i + \sum_{j=1}^{M} a^c_j \tag{16}$$

4. Support Vector Machine-Based Multi-Layer Task Classification

This section addresses the proposed SVM-based multi-class algorithm for weighting each task. The result of this stage is used together with Algorithm 1 to determine the priority of the tasks. The proposed SVM-based algorithm determines the weight of the task into three categories, "high", "med", and "norm", according to two features, where the first one is the patient profile, which represents the medical history of the patient, and the second one is the symptom, which is related to the sensor's response to an action. Figure 2 shows the transitions between states of the weights for the three classes.

1. High: This state indicates that the notification from one of the sensors has one of the symptoms labeled risky, in addition to the fact that the patient has an illness history within their profile; additionally, the received symptom is directly connected to the patient's medical case.
2. Medium: This state includes two cases: The first one implies that the notification has one of the risky symptoms but the patient's medical profile is marked as healthy and has no illness history; this case is represented by (01). The second case is when the patient is labeled as having one of the chronic diseases which requires constant surveillance and the patient has no symptoms at the moment; this case is represented by (10).
3. Normal: This state refers to the situation where all incoming notifications are within safe limits, such as periodic readings, with a clear illness history for the patient.

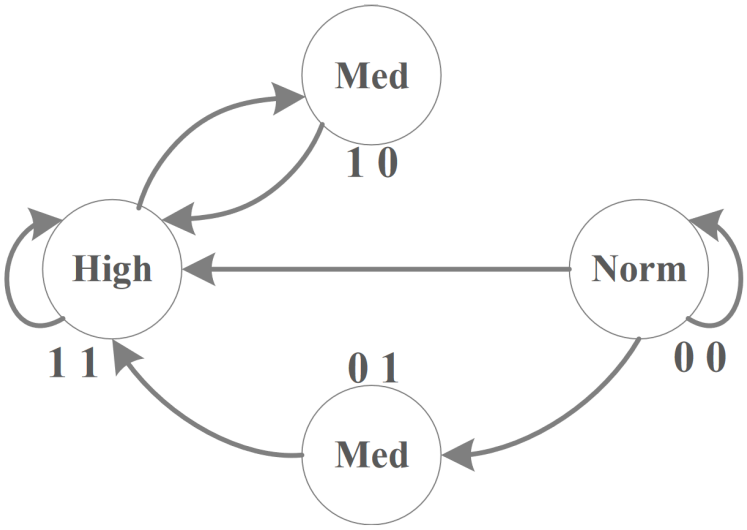

Figure 2. State diagram for task weight determination.

It is worth to highlight here that three classes are sufficient and efficient in targeting low latency for the high class and offering sufficient coverage and connectivity for the normal class. The medium class is classified as the transitional class; it is distinguished from the normal class and is potentially classified as high class once the extracted features tend to be more "critical", e.g., more severe symptoms are detected. To increase classification accuracy among the three classes, three classifiers are required for classifying high/med, med/norm, and high/norm [30].

Highly nonlinear and heterogeneous features lead the task priority determination in this paper to require machine learning-based algorithms [19]. Given the need for machine learning algorithms, SVM, well known for its capability of handling classification in nonlinear solution spaces with fast computation [30], is adopted as the backbone algorithm for task priority determination. Regarding the heterogeneous features considered in task priority determination, new kernel fusing cross-correlation, convolution, and auto-correlation are developed.

In the same manner as in Algorithm 1, the machine learning algorithm SVM is used in Algorithm 2 to classify FNs into three levels (high, medium, and low) based on three aspects: the first one is the physical characteristics of the FNs, which include MIPS, RAM, storage, and the number of CPUs; the second aspect reflects network communication features up/downlink between the FNs and the physical layer; and finally, the third aspect refers to the availability of the resources in each FN and their ability to receive and process new tasks. This procedure is periodically employed to scan the FNs in the service area and update the SVM value in Algorithm 2 to classify the FNs into three levels to meet the latency limits for critical tasks.

Task priority determination, together with fog/cloud computing ranking, which will be described in Section 5, contributes to Algorithm 3, i.e., resource mapping, for efficiently supporting critical tasks in e-Health with low latency and high QoS while ensuring effective connectivity and computation for normal tasks (e.g., monitoring). The resource-mapping algorithm is further illustrated in detail in Section 5.3.

Algorithm 2 Resource classification algorithm.

/* Scan for the available fog node A_N^f */
for $n = 0, 1, 2 \ldots, N$ **do**
    /* 1.0 SVM classify computational capacity (CC) */
    $$SVM\begin{Bmatrix} MIPS \\ RAM \\ CPUs \\ Storage \end{Bmatrix} \Rightarrow (CC(A_n^f))$$
    /* 1.1 SVM classify connection features (CF) */
    $$SVM\begin{Bmatrix} uplink \\ distance \end{Bmatrix} \Rightarrow (CF(A_n^f))$$
    /* 1.2 The Availability of the fog node (A) */
    $$Availabl\begin{Bmatrix} \text{Task spots} \\ \text{available resources} \end{Bmatrix} \Rightarrow (A(A_n^f))$$
end for
/* 2.0 Classify and order fog nodes using SVM */
$$SVM\begin{Bmatrix} (CC(A_n^f)) \\ (CF(A_n^f)) \\ (A(A_n^f)) \end{Bmatrix} \Rightarrow$$
$$\left\{(A_n^f)_{High}, \quad (A_n^f)_{Medium}, \quad \ldots, \quad (A_n^f)_{Low}\right\}$$

Algorithm 3 Resource-mapping algorithm.

/* 1.0 Receiving prioritized fog nodes for Algorithm 2 */
$\left\{(A_n^f)_{High}, \quad (A_n^f)_{Medium}, \quad \ldots, \quad (A_n^f)_{Low}\right\}$
/* 2.0 Receiving prioritized tasks for Algorithm 1 */
prioritized task= $\left\{(x_{high}^{f_n}), \quad (x_{med}^{f_n}), \quad (x_{low}^{f_n})\right\}$
/* checking task's parameters */
/* 2.1 checking the value of task's Payload */
if $PL(x_{k_u}^u) >= PL(high)$ **then**
    $weight(PL(x_{k_u}^u)) == high$
else if $PL(medium) =< PL(x_{k_u}^u) =< PL(high)$ **then**
    $weight(PL(x_{k_u}^u)) == medium$
else
    $weight(PL(x_{k_u}^u)) == low$
end if
/* 3.0 The Orchestrator maps & offloads tasks to *FNs* or *CNs* */

$$\begin{bmatrix} (A_1^f) \\ (A_2^f) \\ (A_n^f) \\ \vdots \\ (A_N^f) \\ \vdots \\ (A_m^c) \\ (A_M^c) \end{bmatrix} \Leftarrow \begin{bmatrix} (x_1^{f_1}) \\ (x_1^{f_1}) \\ (x_k^{f_n}) \\ \vdots \\ \vdots \\ (x_{K-k}^{f_{N-n}}) \\ \vdots \\ (x_K^{f_N}) \end{bmatrix}$$

/* 4.0 The connection capacity */
if $D(_{x_{k_u}}^u) < \tau_{(comm)}(x_{k_u}^u) + \tau_{(wait)}(x_{k_u}^u) + \tau_{(proc)}(x_{k_u}^u)$ **then**
    $(\theta_{x_{k_u}}^u)$ = alarm
    Forward $x_{k_u}^u \Rightarrow$ Cloud node
end if

4.1. Feature and Database Determination

The data used for SVM classification algorithms are real open-source data for scientific research purposes; here, a dataset of heart attack data from [31,32] is used for the SVM algorithm as a critical application, taking into account many parameters, such as the type of chest pain, blood pressure, and cholesterol levels. Using the dataset, the parameters were identified as symptoms for each of the diseases, which helped determine the thresholds for SVM in Algorithm 1 to prioritize tasks.

On the other hand, the datasets used in Algorithm 2 are obtained through our experiments. The used parameters are extracted from the tests in the iFogSim simulator and input into SVM (Algorithm 2). The used parameters include MIPS, RAM, storage, the number of CPUs, connection features, and the available resources. These parameters are used to train the data to classify the FNs.

Both algorithms incorporate these carefully selected features to ensure accurate task classification. We employed five-fold cross-validation for robust model validation and to confirm the effectiveness of these features in practical scenarios.

4.2. New Kernel Design and Margin Maximization

It is commonly recognized that conventional kernels (e.g., linear, quadratic, etc.) cannot effectively be applied to ubiquitous applications [30]. Therefore, in this section, we develop a new type of kernel for enhancing classification accuracy. Conventional kernels and cross-correlation kernels are fused, reflecting symmetric features and anti-symmetric features, respectively, to enhance classification accuracy.

As mentioned previously, three classes, normal, medium, high, represented by C_0, C_1, C_2, respectively, are developed. For each class, q_{max} tasks are considered for the supervision of each class, derived as $X_{C_i} = \{x_{1,C_i}, \ldots, x_{q_{max},C_i}\}$, $i = \{0,1,2\}$. Suppose that L features are considered for each task; then, the similarity between tasks x_{q,C_i} and x_{q',C_j}, $i,j = 0,1,2$, $S_{x_{q,C_i}}^{x_{q',C_j}}$, is defined as a cross-correlation [30]:

$$S_{x_{q,C_i}}^{x_{q',C_j}}(l) = \sum_{n=-\infty}^{+\infty} x_{q,C_i}(n) \cdot x_{q',C_j}(n-l) \tag{17}$$

In general, tasks classified in the same class should gain higher value of $S_{x_{q,C_i}}^{x_{q',C_j}}$, with higher similarity compared with tasks in different classes.

Similarly, convolution and auto-correlation (auto-correlation only for tasks in the same class) [30], representing reversed similarity and self-similarity, respectively, are derived as

$$R_{x_{q,C_i}}^{x_{q',C_j}}(l) = \sum_{n=1}^{L} x_{q,C_i}(l) \cdot x_{q',C_j}(l-n) \tag{18}$$

$$SS_{x_{q,C_i}}^{x_{q',C_i}}(l) = \sum_{n=1}^{L} x_{q,C_i}(n) \cdot x_{q',C_i}(n-l) \tag{19}$$

By nature, cross-correlation reflects the similarity of two tasks, while convolution enhances the accuracy of similarity determination, reversely. Auto-correlation is also adopted, further improving classification performance in sensitivity and specificity.

In this paper, the information obtained by $S_{x_{q,C_i}}^{x_{q',C_j}}$, $R_{x_{q,C_i}}^{x_{q',C_j}}$, and $SS_{x_{q,C_i}}^{x_{q',C_i}}$ is used to model the kernels of task classification, which are derived based on the kernel matrix:

$$\mathbf{K}_{q,q'}^{S} = \begin{bmatrix} K_{1,1}^{S} & \cdots & K_{1,3q_{max}}^{S} \\ \vdots & \ddots & \vdots \\ K_{3q_{max},1}^{S} & \cdots & K_{3q_{max},3q_{max}}^{S} \end{bmatrix} \tag{20}$$

$$\mathbf{K}_{q,q'}^{R} = \begin{bmatrix} K_{1,1}^{R} & \cdots & K_{1,3q_{max}}^{R} \\ \vdots & \ddots & \vdots \\ K_{3q_{max},1}^{R} & \cdots & K_{3q_{max},3q_{max}}^{R} \end{bmatrix} \tag{21}$$

$$\mathbf{K}_{q,q'}^{i,SS} = \begin{bmatrix} K_{1,1}^{i,SS} & \cdots & K_{1,q_{max}}^{i,SS} \\ \vdots & \ddots & \vdots \\ K_{q_{max},1}^{i,SS} & \cdots & K_{q_{max},q_{max}}^{i,SS} \end{bmatrix}, i = \{0,1,2\} \tag{22}$$

where

$$K_{q,q'}^{S} = \sum_{n=1}^{L} \omega_S(n) \cdot S_{x_{q,C_i}}^{x_{q',C_j}}(n) \tag{23}$$

$$K_{q,q'}^{R} = \sum_{n=1}^{L} \omega_R(n) \cdot R_{x_{q,C_i}}^{x_{q',C_j}}(n) \tag{24}$$

$$K_{q,q'}^{i,SS} = \sum_{n=1}^{L} \omega_{SS}(n) \cdot SS_{x_{q,C_i}}^{x_{q',C_i}}(n), i = \{0,1,2\} \tag{25}$$

The weighting of features, i.e., $\{\omega_S(n), \omega_R(n), \omega_{SS}(n)\}$, should be determined according to the demand of applications. For instance, weighting for the feature related to "patients' medical record" should be designed as relatively higher, to potentially increase the specificity of defined critical tasks classified with high priority.

Mercer kernels (i.e., inner product kernels, $K^{S}_{q,q'}$, $K^{R}_{q,q'}$, $K^{SS}_{q,q'}$) obey the Mercer theorem, which satisfies the symmetric and positive semi-definite requirements [30]. Therefore, the kernel for task classification is defined as the sum of the inner product kernels:

$$K^{o}_{q,q'} = \mu_S \cdot K^{S}_{q,q'} + \mu_R \cdot K^{R}_{q,q'} + \sum_{i=0}^{2} \nu_{i,SS} \cdot K^{i,SS}_{q,q'} \tag{26}$$

where $\{\mu_S, \mu_R, \mu_{SS}, \nu_{i,SS}\}, i = \{0,1,2\}$, are weights for the defined inner product kernels. $K^{o}_{q,q'}$, as the sum of Mercer kernels, is also a Mercer kernel, which fuses the strengths of cross-correlation, convolution, and auto-correlation for enhancements in overall classification accuracy. The maximum margin function of $K^{o}_{q,q'}$ is derived as

$$\max \tilde{M}(\boldsymbol{\alpha}) = \sum_{q=1}^{q_{max}} \alpha_q + \frac{1}{2} \sum_{q=1}^{q_{max}} \sum_{q'=1}^{q_{max}} \alpha_q \alpha_{q'} y_q y_{q'} K^{o}_{q,q'} \tag{27}$$

subject to

$$\forall q \in [1, q_{max}], \alpha_q \geq 0 \tag{28a}$$

$$\sum_{q=1}^{q_{max}} \alpha_q y_q = 0 \tag{28b}$$

where α is the Lagrange multiplier and $y \in \{0,1\}$ is the output of classification, with 0 and 1 representing the binary side of each classifier. $\tilde{M}(\boldsymbol{\alpha})$ is developed for all the classifiers and its maximization optimizes overall accuracy in task classification.

Figure 3 clarifies the margins to detect one of the classified (high, medium, and low) cases, when a new notification is generated. The "high" case is highlighted in the middle of the figure with blue color; it represents the matching between high-risk symptoms which are connected to the patient's medical history (patient profile). On the other hand, the other two cases, "medium" and "low", highlighted in red color on either side of the blue zone, reflect either that the present symptoms are not dangerous under normal limits or that the following notifications are normal for those without a previous history of diseases.

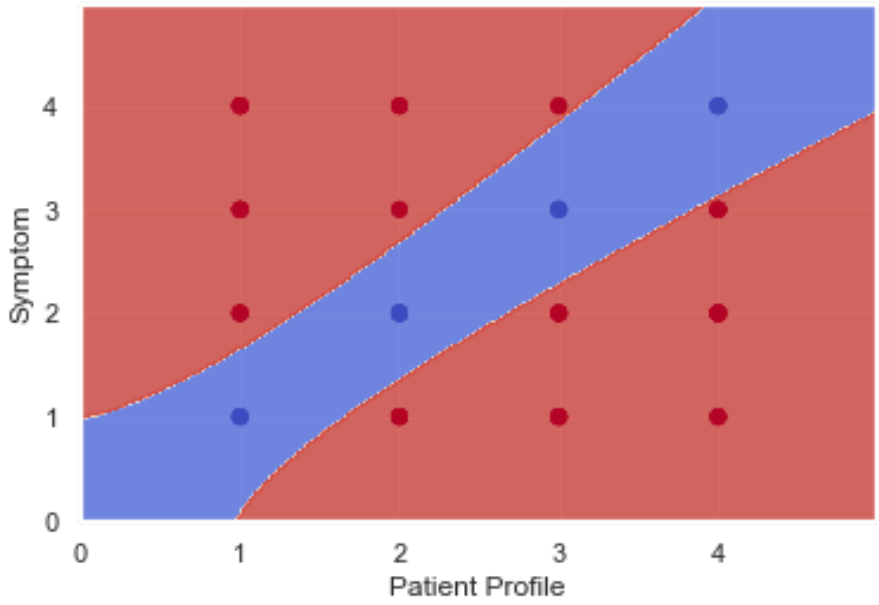

Figure 3. Training dataset and margins for the SVM-based task-scheduling algorithm.

5. Task Scheduling Based on Resource Mapping

In this section, we delve into the intricate process of task scheduling through resource mapping. We introduce three pivotal algorithms that collaborate to ensure efficient task prioritization, resource classification, and the effective mapping of tasks to the most suitable resources. Algorithm 1 focuses on determining the priority of tasks based on various factors, including patient profiles and symptoms. Algorithm 2 classifies fog nodes (FNs) based on their capacity and availability, setting the stage for optimal task allocation. Finally, Algorithm 3 oversees the mapping and offloading of tasks to the appropriate fog or cloud nodes, ensuring that each task is handled by the most fitting resources. Figure 4 shows the collaborative execution and compatibility of these algorithms to serve the proposed priority-based task-scheduling and resource allocation framework.

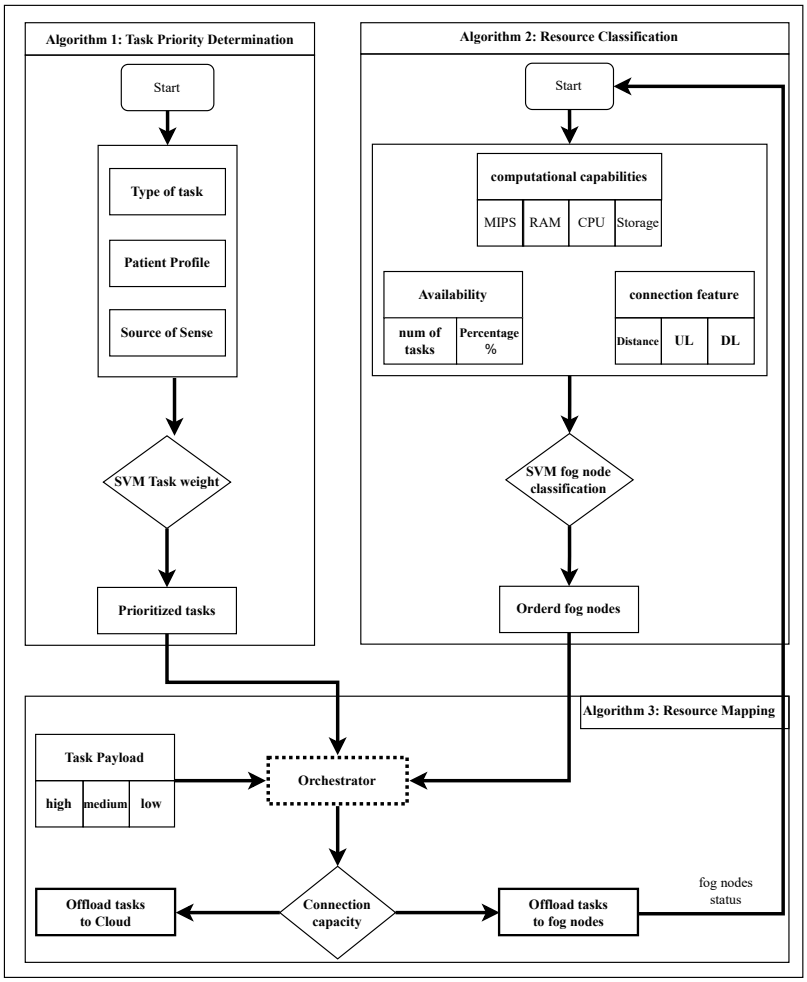

Figure 4. Flowchart showing collaborative execution of the three algorithms.

5.1. Task Priority Determination Algorithm

The task priority determination algorithm (Algorithm 1) contains the following steps, where the orchestrator is responsible for extracting the information from each incoming task:

1. Algorithm 1 checks the values of two fields (patient profile and symptom) in addition to other parameters, such as the payload of the task. If the two values of patient profile and symptom are equal, the SVM_weight of the task is high. On the contrary, if the values of patient profile and symptom are not equal, the SVM_weight of the task is medium. The case of SVM_weight equal to low is when the patient's health record is labeled healthy and the symptom field contains vital indicators in the normal limits.
2. The task's priority value is assigned based on the previously mentioned values.
3. The value of the field type of task is assigned based on the task's priority value.

4. The tasks are ordered in descending order based on their priority.
5. The prioritized-labeled tasks are sent to the orchestrator to be distributed to the proper FNs.

5.2. Resource Classification Algorithm

The resource classification algorithm (Algorithm 2) is a recursive procedure responsible for classifying FNs based on their capacity and availability and includes the following steps:

1. The FN computational capacity, which includes MIPS, RAM, storage, and the number of CPUs and their capacity, is extracted.
2. The topology of the service area is scanned to determine the characteristics of the connection, including the uplink/downlink bandwidth and the distance between the FNs and the devices that should connect to them; this distance is divided into three levels (near, medium, and far) based on the area where the device is located.
3. The FN sends the processing occupancy percentage, i.e., the volume of resources occupied in favor of processing tasks and the percentage of resources available to process new tasks.
4. According to the previous parameters, using the SVM algorithm, the FNs are classified and ordered in descending order into three levels: high, medium, and low.
5. The order of the classified FNs is sent to the orchestrator.

The output of Algorithm 2 is a set of FNs classified and ordered based on their capacity and availability. The capacity is considered a fixed attribute related to an FN's physical characteristics. In contrast, availability is treated as a dynamic attribute, reflecting the current resource usage within each FN. Algorithm 2 performs periodic assessments of the FNs, considering not only their computational capacity and connectivity features but also the processing occupancy percentage. The resulting order of these classified FNs is then communicated to the orchestrator for further task distribution and resource mapping processes.

5.3. Resource-Mapping Algorithm

Algorithm 3 involves the orchestrator mapping tasks to the appropriate FNs based on the classifications provided by Algorithms 1 and 2 through the following steps:

1. The orchestrator receives the classified fog nodes from Algorithm 2.
2. The orchestrator receives the prioritized tasks from Algorithm 1.
3. It checks the value of the payload field and assigns it the label high or medium based on the SVM threshold.
4. The orchestrator maps and offloads tasks to the FNs or CNs based on priority and classification.
5. The orchestrator checks if the network connection capacity is sufficient to serve the incoming requests to meet the latency requirement. If not, the type of task $(\theta^{u}_{x_{k_u}})$ field is labeled with an alarm and forwarded to the cloud node.

In summary, the integrated use of these algorithms forms an effective strategy for managing task scheduling and resource mapping in complex computing environments. They collectively enhance resource utilization, prioritize critical tasks, and ensure optimal task distribution, leading to improved system performance and efficiency.

5.4. Complexity Analysis

The computational complexity of the three algorithms is as follows:

- Algorithm 1 task priority determination algorithm: The complexity of this algorithm is primarily dependent on the number of tasks. If N represents the total number of tasks, then the complexity is $O(N)$, as each task requires a constant amount of time for processing.
- Algorithm 2 resource classification algorithm: The complexity is influenced by the number of fog nodes, denoted by M. Since each node is classified independently, the algorithm exhibits linear complexity, $O(M)$.

- Algorithm 3 resource-mapping algorithm: This algorithm combines aspects of both task prioritization and resource classification. With N tasks and M fog nodes, the worst-case complexity could be $O(N \times M)$, particularly in scenarios where each task must be considered for every node.

5.5. Offloading Scheme

This section explains the collaborative execution of Algorithms 1–3 in order to effectively offload the ordered tasks to the corresponding fog nodes and cloud nodes.

As mentioned in Section 5, the prioritized task list from Algorithm 1 and the classified FNs from Algorithm 2 are then used as input for Algorithm 3. In this algorithm, the orchestrator maps the highest-priority tasks to the available FNs with the highest capacity. Tasks with medium priority are assigned to FNs classified as having medium capacity, and low-priority tasks are offloaded to FNs with a low classification.

The task distribution process is a continuous and dynamic operation managed by Algorithm 3. The orchestrator periodically scans all existing FNs, assessing their current capacity and availability. After each assessment round, the FNs are reordered based on their updated capacity and availability status, ensuring the most efficient utilization of resources for processing the remaining tasks.

Figure 5 shows the change in the order of the FNs after each round of scanning, where the FNs that were originally classified as high are placed at the end of the ordered FN list; this is due to the lack of available resources in these FNs, as these FNs are busy handling other tasks. The other FNs that were originally labeled medium are moved to be classified as high among the available FNs. Figure 5 is plotted to clarify the reordering and transformation process of the FNs and the received tasks. In Figure 5, state = 1 shows that the high-priority tasks were sent to the FNs labeled high, e.g., tasks ID = 14, ID = 75, and ID = 1 were assigned to FNs ID = 14, ID = 2, ID = 10, respectively, and the tasks with medium priority were sent to the FNs labeled medium. Tasks and FNs with a low classification are handled in the same manner. In the next round, as shown in Figure 5 (state = state + 1), we can notice that FN ID-14 is placed with the low-class nodes based on its weak availability to receive and process new tasks. In the same figure, we notice that there is a new FN (ID-13) listed in the high class due to its capacity and availability and that FN ID-23, which was primarily classified as a medium-capacity FN, has been reallocated in a new spot in the high-capacity class.

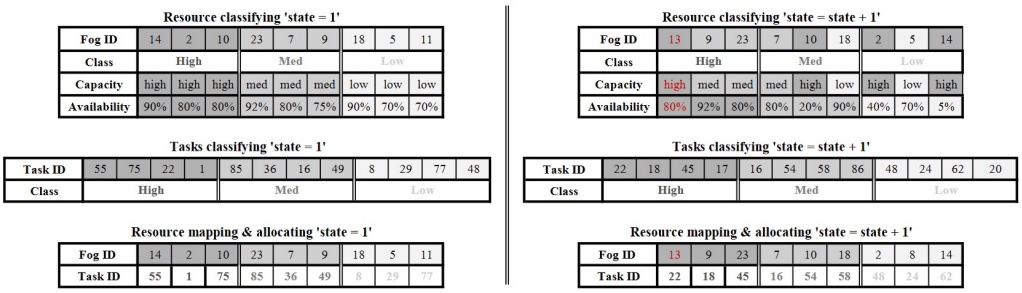

Figure 5. Resource allocation recursive procedure.

6. Performance Analysis

The performance of the proposed algorithm was evaluated by simulating task scheduling and resource mapping and allocating in an iFogSim simulator. In our simulations, the results of our algorithm were compared to those of the widely used first-come, first-served (FCFS) algorithm. FCFS has been well acknowledged in recent works, and its framework represents a widely represented state-of-the-art method for network-level resource mapping and allocation [33–35]. The simulated scenarios considered modifying the number of clusters, FNs per cluster, and their connected IoT devices. The values of the parameters (CPU, RAM, bandwidth, etc.) of the fog devices and the tasks were selected randomly among certain groups of simulation settings, as tabulated in Table 2. A tree-based network

topology was used in the simulated scenarios, where the number of fog devices and related sensors is equal in each cluster. As a starting stage, many bio-potential signals from patients were frequently captured and analyzed. Then, we compared the analyzed data with the database that contains the health records of the patients in order to detect any critical bio-potential.

Table 2. Summary of experimental configuration.

Element	Parameter	Units	Value
Cloud	CPU	MIPS	44,800
	RAM	MB	40,000
	Uplink	bytes/ms	20,000
	Downlink	bytes/ms	20,000
Fog device	CPU	MIPS	{2048, 1024, 768, 512, 256}
	RAM	MB	{2048, 1024, 768, 512, 256}
	Uplink	bytes/ms	{8000, 4000, 2000}
	Downlink	bytes/ms	{8000, 4000, 2000}
Task	CPU length	MIPS	{2000, 1000, 700, 500, 200}
	Network length	bytes	{4000, 2000, 1000}

6.1. Execution Time

Figure 6 presents the difference between the proposed task scheduling/resource mapping and allocation and the built-in scheduling algorithm. We can notice that in the case of Figure 6a, which has one cluster, the difference in execution time between the two algorithms remains close when the number of tasks is 40 or 80, respectively; when the number of tasks becomes large (120 tasks), the difference between the two execution times in the case of one cluster becomes clear in favor of our proposed algorithm.

In the four cases (a, b, c, and d) illustrated in Figure 6, we can notice that increasing the number of the tasks per cluster, in the case of the built-in algorithm, leads to an exponential increase in execution time. This increase in execution time is clearly visible when there is a large number of tasks, in contrast to the proposed algorithm, which keeps the increase in execution time in direct proportion to the increase in the number of tasks per cluster. The proposed algorithm is able to achieve an average execution time for critical tasks of 0.2393 ms, and for non-critical/normal task, it achieves an average execution time equal to 0.5001 ms. In the case of utilizing the FCFS algorithm with the same architecture, we can notice that the average execution times for the critical tasks and normal tasks are 0.9162 ms and 1.0419 ms, respectively. Hence, the proposed algorithm is able to achieve better execution time for all cases (critical/normal) compared with the FCFS algorithm, which has almost similar average execution time values.

6.2. Latency

Latency is one of the main KPIs (key performance indicators) to be considered when implementing a real-time healthcare system and has to be reduced to achieve the required high efficiency. In the fog–cloud architecture, using FNs reduces latency and enhances the overall execution time by processing tasks in the FNs locally, utilizing the available resources and decreasing the number of tasks that should be transmitted to and handled by the cloud. For the purpose of emphasizing the efficiency of the proposed algorithm, a comparison of latency is evaluated in both cloud-only and fog–cloud architectures, where the variable factor is the number of connected devices. In the cloud-only architecture, we can notice that the increase in the number of sensors directly leads to a steady increase in latency. Moreover, when the number of connected devices is above 45, the latency starts increasing in a more rapid way from 224.91 ms and reaches 331.81 ms when the connected devices are 60, as illustrated in Figure 7. Contrarily, in the fog–cloud architecture, increasing the number of devices has a limited effect on the achieved delay. When varying the number of connected devices from 20 to 60, the delay limit only varies between 11.4 ms and

23.06 ms, achieving an almost 90% reduction in delay compared with the cloud-only architecture, as highlighted in Figure 7.

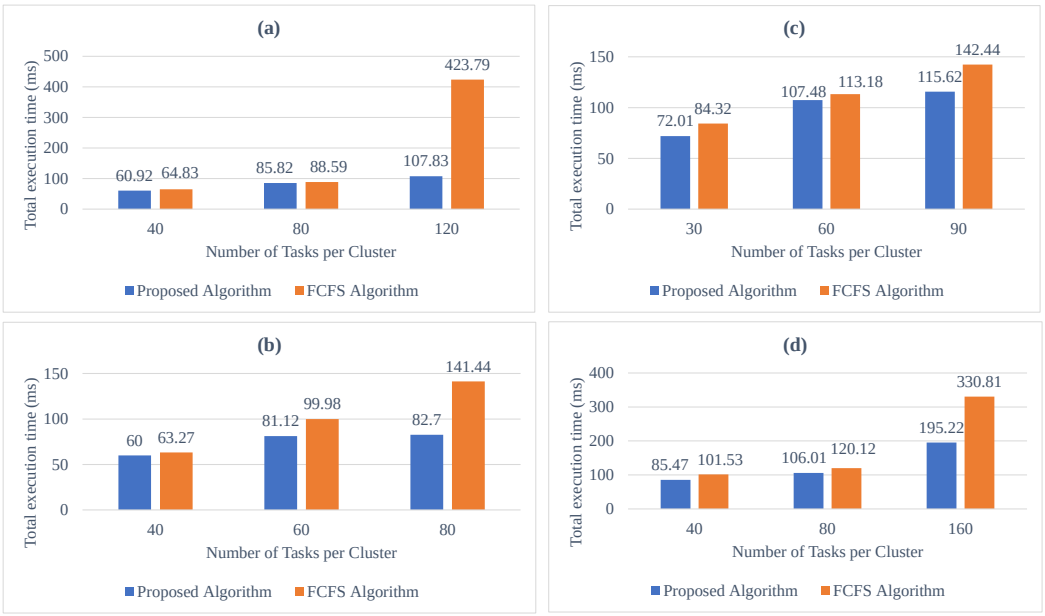

Figure 6. Total execution time in different clusters: (**a**) one cluster; (**b**) two clusters; (**c**) three clusters; (**d**) four clusters.

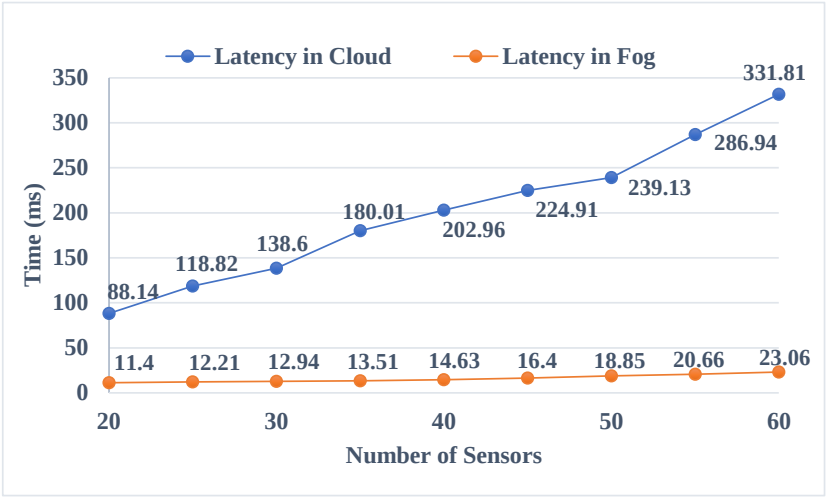

Figure 7. Comparison of latency.

6.3. Network Utilization

Regarding the network utilization efficiency aspect, which is measured in KByte per second, Figure 8 highlights the total network usage for four different cases (one cluster, two clusters, three clusters, four clusters). Each cluster is tested for three various groups of tasks (40, 80, 120), applied with FCFS-based algorithms (averaged based on [34,35], representing recent works related to machine learning-based FCFS and hierarchical FCFS, respectively) and the proposed algorithm. In the case of one, two, and three clusters, network usage in both scenarios is almost convergent, and our proposal attains lower network usage.

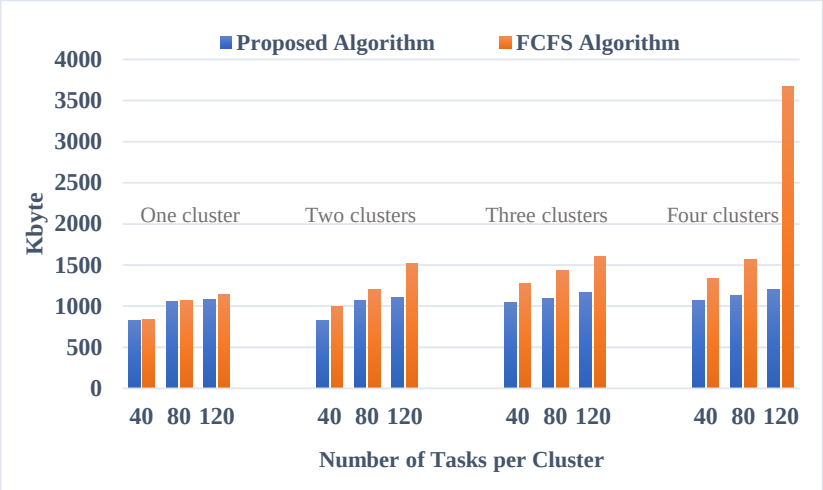

Figure 8. Network usage among different clusters.

Specifically, our proposed algorithm achieves better results when the number of tasks and clusters is high. In the last case, the FCFS algorithm results in about four times larger network usage compared with our work.

Consequently, the increase in network usage affects the total execution time, waiting time, and communication time. Furthermore, it also results in increased costs of network usage and power consumption. Concretely, comparing the results achieved in [34] for solving task scheduling with a time–cost-aware scheduling (TCaS) algorithm and the performance of our proposed solution, it can be seen that when the number of tasks is more than 100 in each cluster with five FNs, our proposed algorithm achieves better execution time. When the number of tasks is 120, 160, or 180, our solution achieves a delay of 80 ms, 83 ms, or 105 ms, whereas TCaS produces a delay of almost 150 ms, 195 ms, or 230 ms, respectively. Additionally, authors of the work in [35] proposed a method to reduce latency and network consumption in a remote pain-monitoring system. Comparing the achieved results, the delay obtained with our algorithm when the number of sensors is 20, 30, 40, or 50 in a cloud-only architecture is 88 ms, 138 ms, 202 ms, or 239 ms compared with 215 ms, 225 ms, 233 ms, or 238 ms in the cited paper, respectively. While in a fog–cloud architecture, our solutions produce slightly better values, especially when the number of sensors is high.

7. Conclusions

In this paper, we propose a fog–cloud hierarchical task-scheduling scheme for e-Health applications. Different from conventional task scheduling, we formulate features of tasks comprehensively, fusing network-level and service-level parameters simultaneously, which are further considered in the proposed support vector machine (SVM)-based task classification algorithm. Classified tasks are assigned priorities, giving guidance to be allocated to proper fog/cloud nodes, for network utilization efficiency maximization and overall latency reduction. In particular, the proposed task-scheduling scheme is capable of effectively achieving latency minimization for critical tasks as defined based on the demand of both networks and services. Simulation results show that the proposed algorithm was able to minimize the total execution time for all tasks and especially for the critical ones. The integration of SVM enhanced the latency and network usage in parallel with the increased number of tasks.

The current use of SVM algorithms limits the model's performance in complex datasets with higher dimensions to comprehensively serve ubiquitous healthcare applications. Future work will focus on integrating advanced algorithms, including deep learning and hybrid models combining SVM with other techniques, to discuss their feasibility with comprehensive consideration of computational complexity and algorithmic efficiency simultaneously.

Author Contributions: Conceptualization, A.R.; methodology, A.A. and H.R.C.; software, A.A.; validation, A.A. and H.R.C.; formal analysis, all authors; investigation, all authors; data curation, all authors; writing—original draft preparation, A.A. and H.R.C.; writing—review and editing, A.R. and R.L.A.; supervision, A.R. and R.L.A.; project administration, A.R. and R.L.A.; funding acquisition, A.R. and R.L.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020-UIDP/50008/2020.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

A_M^c	Cloud node
A_N^f	Fog node
U_{IoT}	IoT device
$x_{K_u}^u$	Task
$C(x_{k_u}^u)$	Cost
$P(x_{k_u}^u)$	Priority of a task
$D(x_{k_u}^u)$	Latency of a task
$L(x_{k_u}^u)$	Payload of a task
$\theta(x_{k_u}^u)$	Type of task
$\beta(x_{k_u}^u)$	Patient profile
$\mu(x_{k_u}^u)$	Patient preliminary symptoms
δ_u^f	Distance between an IoT device and a fog node
δ_u^c	Distance from a fog node to the cloud
$\tau_{(comm)}$	Communication time of a task
$\tau_{(wait)}$	Waiting time of a task
$\tau_{(proc)}$	Processing time of a task
$t_{(trans)}$	Transmission delay
R_{cloud}, R_{fog}	Data rate of a fog node and that of a cloud node
$W_{a_M^c}, W_{a_N^f}$	Link bandwidth of a fog node and that of a cloud node
$SINR$	Signal-to-interference-plus-noise ratio
$r_{u_{(IoT)}}^{UL}$	Uplink transmitting rate
$\psi(x_{k_u}^u)$	Processing time of a task
$\Gamma(A_n^c), \Gamma(A_m^f)$	Computing capacity of a fog node and that of a cloud node
T_u^z	Total number of time slots in a processing node
$rq_x^u(t)$	Required resources for a task in a given slot time

References

1. Liu, J.; Ahmed, M.; Mirza, M.A.; Khan, W.U.; Xu, D.; Li, J.; Aziz, A.; Han, Z. RL/DRL Meets Vehicular Task Offloading Using Edge and Vehicular Cloudlet: A Survey. *IEEE Internet Things J.* **2022**, *9*, 8315–8338. [CrossRef]
2. Chi, H.R.; Domingues, M.F.; Radwan, A. QoS-aware Small-Cell-Overlaid Heterogeneous Sensor Network Deployment for eHealth. In Proceedings of the 2020 IEEE SENSORS, Rotterdam, The Netherlands, 25–28 October 2020; pp. 1–4. [CrossRef]
3. Chi, H.R. Editorial: Edge Computing for the Internet of Things. *J. Sens. Actuator Netw.* **2023**, *12*, 17. [CrossRef]
4. Kashani, M.H.; Mahdipour, E. Load Balancing Algorithms in Fog Computing. *IEEE Trans. Serv. Comput.* **2023**, *16*, 1505–1521. [CrossRef]
5. Chi, H.R.; Domingues, M.d.F.; Zhu, H.; Li, C.; Kojima, K.; Radwan, A. Healthcare 5.0: In the Perspective of Consumer Internet-of-Things-Based Fog/Cloud Computing. *IEEE Trans. Consum. Electron.* **2023**, 1. [CrossRef]
6. Radwan, A.; Chi, H.R. Towards Cell-Free Networking: Analytical Study of Ultra-Dense On-Demand Small Cell Deployment for Internet of Things. In Proceedings of the 2023 International Wireless Communications and Mobile Computing (IWCMC), Marrakesh, Morocco, 19–23 June 2023; pp. 1202–1207. [CrossRef]
7. Strumberger, I.; Tuba, M.; Bacanin, N.; Tuba, E. Cloudlet Scheduling by Hybridized Monarch Butterfly Optimization Algorithm. *J. Sens. Actuator Netw.* **2019**, *8*, 44. [CrossRef]

8. Mattia, G.P.; Beraldi, R. On real-time scheduling in Fog computing: A Reinforcement Learning algorithm with application to smart cities. In Proceedings of the 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), Pisa, Italy, 21–25 March 2022; pp. 187–193. [CrossRef]
9. AlZailaa, A.; Chi, H.R.; Radwan, A.; Aguiar, R. Low-Latency Task Classification and Scheduling in Fog/Cloud based Critical e-Health Applications. In Proceedings of the ICC 2021-IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6. [CrossRef]
10. Semmoud, A.; Hakem, M.; Benmammar, B.; Charr, J.C. Load balancing in cloud computing environments based on adaptive starvation threshold. *Concurr. Comput. Pract. Exp.* **2020**, *32*, e5652. [CrossRef]
11. Benblidia, M.A.; Brik, B.; Merghem-Boulahia, L.; Esseghir, M. Ranking Fog nodes for Tasks Scheduling in Fog-Cloud Environments: A Fuzzy Logic Approach. In Proceedings of the 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 24–28 June 2019; pp. 1451–1457. [CrossRef]
12. Abdel-Basset, M.; El-Shahat, D.; Elhoseny, M.; Song, H. Energy-Aware Metaheuristic Algorithm for Industrial-Internet-of-Things Task Scheduling Problems in Fog Computing Applications. *IEEE Internet Things J.* **2021**, *8*, 12638–12649. [CrossRef]
13. Hosseini, E.; Nickray, M.; Ghanbari, S. Optimized task scheduling for cost-latency trade-off in mobile fog computing using fuzzy analytical hierarchy process. *Comput. Netw.* **2022**, *206*, 108752. [CrossRef]
14. Tuli, S.; Basumatary, N.; Gill, S.S.; Kahani, M.; Arya, R.C.; Wander, G.S.; Buyya, R. HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments. *Future Gener. Comput. Syst.* **2020**, *104*, 187–200. [CrossRef]
15. Azizi, S.; Shojafar, M.; Abawajy, J.; Buyya, R. Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach. *J. Netw. Comput. Appl.* **2022**, *201*, 103333. [CrossRef]
16. Kanbar, A.B.; Faraj, K. Region aware dynamic task scheduling and resource virtualization for load balancing in IoT–Fog multi-cloud environment. *Future Gener. Comput. Syst.* **2022**, *137*, 70–86. [CrossRef]
17. Okegbile, S.D.; Maharaj, B.T.; Alfa, A.S. A Multi-User Tasks Offloading Scheme for Integrated Edge-Fog-Cloud Computing Environments. *IEEE Trans. Veh. Technol.* **2022**, *71*, 7487–7502. [CrossRef]
18. Mutlag, A.A.; Khanapi Abd Ghani, M.; Mohammed, M.A.; Maashi, M.S.; Mohd, O.; Mostafa, S.A.; Abdulkareem, K.H.; Marques, G.; de la Torre Díez, I. MAFC: Multi-Agent Fog Computing Model for Healthcare Critical Tasks Management. *Sensors* **2020**, *20*, 1853. [CrossRef] [PubMed]
19. Chakraborty, C.; Mishra, K.; Majhi, S.K.; Bhuyan, H.K. Intelligent Latency-Aware Tasks Prioritization and Offloading Strategy in Distributed Fog-Cloud of Things. *IEEE Trans. Ind. Inform.* **2023**, *19*, 2099–2106. [CrossRef]
20. Gupta, S.; Iyer, S.; Agarwal, G.; Manoharan, P.; Algarni, A.D.; Aldehim, G.; Raahemifar, K. Efficient Prioritization and Processor Selection Schemes for HEFT Algorithm: A Makespan Optimizer for Task Scheduling in Cloud Environment. *Electronics* **2022**, *11*, 2557. [CrossRef]
21. Alatoun, K.; Matrouk, K.; Mohammed, M.A.; Nedoma, J.; Martinek, R.; Zmij, P. A Novel Low-Latency and Energy-Efficient Task Scheduling Framework for Internet of Medical Things in an Edge Fog Cloud System. *Sensors* **2022**, *22*, 5327. [CrossRef]
22. Khosroabadi, F.; Fotouhi-Ghazvini, F.; Fotouhi, H. SCATTER: Service Placement in Real-Time Fog-Assisted IoT Networks. *J. Sens. Actuator Netw.* **2021**, *10*, 26. [CrossRef]
23. Nagarajan, S.M.; Devarajan, G.G.; Mohammed, A.S.; Ramana, T.V.; Ghosh, U. Intelligent Task Scheduling Approach for IoT Integrated Healthcare Cyber Physical Systems. *IEEE Trans. Netw. Sci. Eng.* **2022**, *10*, 2429–2438. [CrossRef]
24. Chen, J.; He, Y.; Zhang, Y.; Han, P.; Du, C. Energy-aware scheduling for dependent tasks in heterogeneous multiprocessor systems. *J. Syst. Archit.* **2022**, *129*, 102598. [CrossRef]
25. Abdelmoneem, R.M.; Benslimane, A.; Shaaban, E. Mobility-aware task scheduling in cloud-Fog IoT-based healthcare architectures. *Comput. Netw.* **2020**, *179*, 107348. [CrossRef]
26. Ali, I.M.; Sallam, K.M.; Moustafa, N.; Chakraborty, R.; Ryan, M.; Choo, K.K.R. An Automated Task Scheduling Model Using Non-Dominated Sorting Genetic Algorithm II for Fog-Cloud Systems. *IEEE Trans. Cloud Comput.* **2022**, *10*, 2294–2308. [CrossRef]
27. Cheikhrouhou, O.; Mershad, K.; Jamil, F.; Mahmud, R.; Koubaa, A.; Moosavi, S.R. A lightweight blockchain and fog-enabled secure remote patient monitoring system. *Internet Things* **2023**, *22*, 100691. [CrossRef]
28. Tong, Z.; Deng, X.; Ye, F.; Basodi, S.; Xiao, X.; Pan, Y. Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment. *Inf. Sci.* **2020**, *537*, 116–131. [CrossRef]
29. Balevi, E.; Gitlin, R.D. Optimizing the number of fog nodes for cloud-fog-thing networks. *IEEE Access* **2018**, *6*, 11173–11183. [CrossRef]
30. Chui, K.T.; Tsang, K.F.; Chi, H.R.; Ling, B.W.K.; Wu, C.K. An Accurate ECG-Based Transportation Safety Drowsiness Detection Scheme. *IEEE Trans. Ind. Inform.* **2016**, *12*, 1438–1452. [CrossRef]
31. Ramani, P.; Pradhan, N.; Sharma, A.K. Classification Algorithms to Predict Heart Diseases—A Survey. In Proceedings of the Computer Vision and Machine Intelligence in Medical Image Analysis, Accra, Ghana, 29–31 May 2019; Gupta, M., Konar, D., Bhattacharyya, S., Biswas, S., Eds.; WikiCFP: Singapore, 2020; pp. 65–71.
32. Kumar, P.; Chauhan, R.; Stephan, T.; Shankar, A.; Thakur, S. A Machine Learning Implementation for Mental Health Care. Application: Smart Watch for Depression Detection. In Proceedings of the 2021 11th International Conference on Cloud Computing, Data Science and Engineering (Confluence), Noida, India, 28–29 January 2021; pp. 568–574. [CrossRef]

33. Mahmud, R.; Pallewatta, S.; Goudarzi, M.; Buyya, R. iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *J. Syst. Softw.* **2022**, *190*, 111351. [CrossRef]
34. Sing, R.; Bhoi, S.K.; Panigrahi, N.; Sahoo, K.S.; Bilal, M.; Shah, S.C. EMCS: An Energy-Efficient Makespan Cost-Aware Scheduling Algorithm Using Evolutionary Learning Approach for Cloud-Fog-Based IoT Applications. *Sustainability* **2022**, *14*, 15096. [CrossRef]
35. Hassan, S.R.; Ahmad, I.; Ahmad, S.; Alfaify, A.; Shafiq, M. Remote Pain Monitoring Using Fog Computing for e-Healthcare: An Efficient Architecture. *Sensors* **2020**, *20*, 6574. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.