Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2021, 8(8):236-244**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Cloud Migration Techniques for Enhancing Critical Public Services: Mobile Cloud-Based Big Healthcare Data Processing in Smart Cities

Premkumar Ganesan

Technology Leader in Digital Transformation for Government and Public Sector, Baltimore, Maryland

Abstract: Interest in "Smart Cities" is growing due to the potential improvements in residents' quality of life. The concept encompasses various areas, including smart communities, smart transportation, and smart healthcare. For Smart City services, particularly Smart healthcare, to make intelligent decisions, it is essential to share and analyze Big Healthcare Data in real-time. Seamless connectivity and access to Smart healthcare services, people, and sensors are vital, achievable only with a robust mobile and wireless communication network. Mobile Cloud Computing (MCC) offers an effective solution for meeting end users' Quality of Service (QoS) requirements by offloading the processing, sharing, and analysis of Big Healthcare Data from mobile devices to cloud resources. Resource migration, or virtual machine (VM) migration, plays a significant role in smart city healthcare. This paper presents a methodology for the collaborative migration of virtual machines in smart city environments, supporting heterogeneous MCC-based smart healthcare systems. The methodology is grounded in Ant Colony Optimization (ACO), addressing the VM migration challenge by leveraging user mobility alongside cloud-provisioned virtual machine resources. Additionally, we provide a thorough performance evaluation to compare our proposed approach with current best practices.

Keywords: Big data, Cloud computing, Mobile cloud, Healthcare.

1. Introduction

The "Smart City" concept, fueled by advancements in ICT, has unlocked new possibilities for enhancing urban living standards. By connecting "smart" objects, people, and sensors, a wide range of services, including smart healthcare, transportation, and communities, can be delivered. Many Smart City services, especially new smart healthcare services, require real-time computation accessible from any location. Key applications in this area include telemedicine, critical patient monitoring, data collection, and personalized healthcare. Given that these healthcare applications generate vast amounts of Big Healthcare Data in real-time, it is crucial to have nearby computational resources readily available. Connecting and accessing these resources with minimal latency is challenging without a robust mobile and wireless communication infrastructure. Mobile Cloud Computing (MCC) addresses this by reducing task execution times and real-time communication latency in Smart City environments. MCC efficiently leverages the processing power, memory, and network resources of distributed cloud servers to run smart healthcare applications on mobile devices. By offloading tasks to powerful cloud servers, MCC accelerates the execution of these applications. Additionally, MCC enables real-time query processing in smart healthcare apps, which is critical for patient care. However, the physical distance between cloud servers and mobile devices can impact the execution time of interactive applications, potentially increasing response times. To mitigate this, cloudlets—local clouds that bring cloud resources closer to the user—have been proposed. By offloading tasks to the nearest cloudlet, users can reduce overall execution time. Virtualization technology allows cloudlet hardware resources to be shared via virtual machines (VMs), which

provide resources such as CPU, RAM, and network bandwidth. Although cloud computing resource provisioning has been extensively studied, maintaining Quality of Service (QoS) in MCC remains challenging due to user mobility. As users move between different Access Points (APs), the distance to the cloudlet may increase, leading to longer task execution times.

To address this issue, we propose a virtual machine migration method for a mixed-mode mobile computing cluster that accounts for user movement patterns. When a user moves to a different cloudlet, the resource or VM should be transferred to the cloudlet nearest to their current location. For instance, consider a visually impaired person using an app that captures and processes images of their environment through a cloudlet. If the user moves away from the current cloudlet, the app's performance may degrade due to slower response times. Integrating a virtual machine migration method into the system can ensure that the cloudlet closest to the user is selected, preventing performance loss.

High levels of human activity, stemming from widespread urbanization, have negatively impacted cities' resilience—their ability to absorb, recover from, and prepare for future economic, institutional, environmental, and social shocks. These challenges have significant environmental and societal implications. The definitions of sustainability and resilience are debated among scholars; some view them as synonymous, while others believe resilience is the foundation of sustainability. Despite these differing perspectives, both sustainability and resilience are critical systemic issues affecting our built environment. To improve urban living conditions, we need smart innovations and new approaches to public health, education, security, and the optimization of energy, waste, and traffic management. The importance of urban resilience in sustainable and smart city planning is greater than ever. A study in Ref. [10] assessed and ranked 187 smart cities in China using a Multi-Criteria Decision-Making (MCDM) method, revealing a lack of resilience. Further research identified a positive correlation between smart cities and urban resilience, suggesting that enhancing city intelligence can improve resilience. Addressing complex sustainability and resilience challenges in smart cities requires integrating multiple technologies, including the Internet of Things (IoT), software, user interfaces, and communication networks. The IoT enables everyday objects like vehicles, sensors, and appliances to communicate and exchange data. Recent developments in industrial applications have embraced the IoT infrastructure, with Cyber-Physical Systems playing a crucial role.

Smart manufacturing and industries are advancing towards creating sustainable smart cities and societies. Data collected by IoT sensors and devices are stored on cloud servers, enabling seamless integration of physical and digital elements in urban settings through data analytics and interconnected devices. This integration fosters more effective collaboration between businesses and governments, benefiting the economy and the city's residents. Smart City software, leveraging data, connectivity, and automation, enhances city operations, services, and residents' quality of life. It typically includes features such as data analytics, IoT, urban mobility, energy management, and citizen engagement platforms.

2. Literature Review

The health industry has recently come to recognize the Internet's potential as a vital tool in the quest for improved patient care and overall quality of life. The ability to analyze and analyze data in real-time through distant servers is one of the many benefits of utilizing the Internet in the health industry. In theory, the idea of "the cloud" may efficiently store and run applications over the Internet [13]. Big data centers provide a service under this model by making available to private enterprises and people as well as government agencies a portion of their hardware and software infrastructure. Without spending a fortune on purchasing, maintaining, and managing them, these clients will get access to computing resources with rising capacity as soon as they pay for the service. The main backbone that enables IoT applications is cloud computing. Hundreds or thousands of devices make up an IoT ecosystem, and they are always sending out requests to have their data analyzed. The IoHT stands for the proliferation of IoT devices utilized for e-health applications within the framework of Healthcare 4.0 [14]. As anticipated, this approach floods a central processing server with requests, which can strain the server's network and need more processing power than a single computer typically has. This is where cloud computing's scalability and pay-as-you-go features come into play as a means of processing data for Internet of Things applications. The increased network latency that occurs when an IoT device sends a query to a cloud server, however, can be intolerable in some cases.

In certain e-health cases, such those involving remote electrocardiograms (ECGs), the timing of data collection and processing is crucial to the proper operation of the system. Our inability to wait for a message to be received, processed, and returned from the cloud is a common occurrence, especially in time-critical applications [15]. In addition, increasing the cloud's capacity to handle a high volume of requests will increase power usage, even in a highly scalable setting.

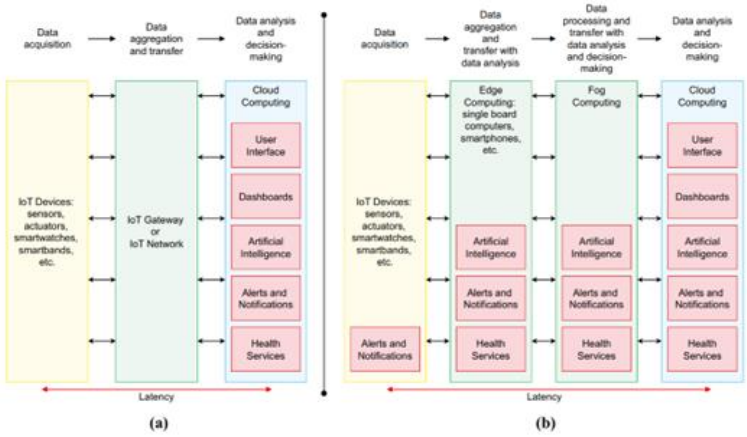

Fig. 1: *Typical methods of health monitoring (a) and VitalSense (b) in comparison. Health services can be accessed from any location via applications.*

It is essential to develop innovative designs and solutions that can manage several devices and requests concurrently while preserving Quality of Service (QoS) in order to make IoT systems more scalable [16]. In line with this statement, fog computing brings data generators closer to the services provided by the conventional cloud model [17]. However, edge computing steps in to help with processing and decision assistance at the network's boundary, close to the IoT device.

The key features of computing in the fog or at the network's edge include low latency, improved support for collecting data from a distributed geographic location, and mobility across a large number of nodes. As a result, applications may run in real-time and device heterogeneity can be better supported with mostly wireless connectivity. In order to prevent network traffic from being delayed due to round-trip delays, the data that the sensors read is first stored in a temporary database before being processed and sent to the cloud. To address cases like Coronavirus disease (COVID-19), it is relevant for healthcare and pandemic research to use an architecture that combines cloud, fog, and the edge [18]. The most critical point is that studies including long-COVID-19 durations are now the standard. The objective is to continuously monitor the vital signs of sick persons before to exposure [19]. Remote health monitoring systems often use a three-tier design to monitor vital signs, as shown in Figure 1(a). Messaging nodes, gateways to the fog, and the cloud make up this design [20]. According to their review of existing efforts in the field, the following problems are not solved all at once:

(i) the ability to track and protect an individual's vital signs and the locations they've been to in a smart city.

(ii) AI to proactively provide health services, benefiting not only hospitals and the public sector but also end-users.

(iii) cutting-edge methods to handle quality of service issues, elastic processing capacity, and a scalable and efficient messenger notification system.

3. Methodology

A. System Architecture

According to our assumptions, the Mobile Cloud Computing (MCC) environment is composed of three tiers, and the backbone network is comprised of a collection of M access points (APs). The first level of the computing infrastructure, known as the master cloud, includes a number of public cloud providers. These providers include Amazon Elastic Compute Cloud (EC2), Google App Engine, and Microsoft Azure. In the

mobile cloud architecture, a group of interconnected cloudlets make up the second layer, often called the backbone layer. The third level, or user layer, encompasses all mobile devices, including smartphones, wearables, and others. Cloud resources that are geographically close to a user are accessible through devices in the third tier. Each access point is associated with a single cloudlet, denoted as C = {C1,C2,C3.....,CL}. Here is a sample network scenario that we have prepared, as shown in Figure 1. The backhaul network connects these cloudlets, and the notation Bi,j represents the bandwidth between cloudlets i and j.

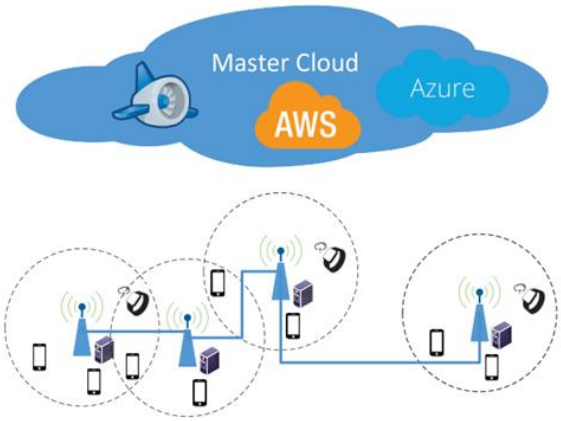

***Figure 2**. The architecture of mobile clouds.*

Control and monitoring of a collection of cloudlets is the responsibility of the master cloud (MC). Each cloudlet has a high-speed network connection to the master cloud, and it transmits data created by its hypervisor to the master clouds. There is a predetermined amount of RAM (Mk) and CPU (S p k) available to a cloudlet server i. The variable V = {V1, V2, V3....., VN} represents the number of virtual machines (VMs) that each cloudlet is obligated to provide. Virtual machines (VMs) allow users to offload tasks to other computers. Furthermore, we take it as read that all users are on the go and that they switch between multiple virtual machines while working on a single task. Our assumption is that there are no interdependencies between those virtual machines (VMs). A task and a virtual machine (VM) will be used interchangeably throughout the rest of this work.

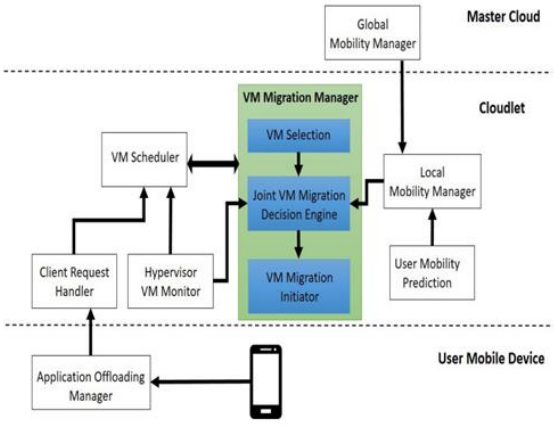

***Figure 3.** System for migrating virtual machines.*

B. Virtual Machine Migration System

Two crucial functions of a computation migration system are shown in Fig. (3). Transferring computations from resource-constrained mobile devices to cloudlets is the first step towards reducing task execution time and over-provisioning of resources. Next, we need to transfer VMs between two cloudlets. The application offloading manager supposedly helps a mobile device figure out which sections of the app can be offloaded and where. The client-request handler of a cloudlet is in charge of receiving requests from mobile devices and sending them on to the virtual machine scheduler so that tasks can be planned for execution. In order for the VM scheduler to

start scheduling virtual machines, the hypervisor supplies it with the required data. The virtual machine migration manager also coordinates with the virtual machine scheduler at regular intervals to migrate and select tasks in an effort to reduce execution times after a specified time epoch. Our primary goal in this project is to provide the VM migration manager with an algorithm that can efficiently remap jobs, choose a group of cloudlets, and reduce execution time. The VM Migration Initiator, Joint VM Migration Engine, and VM Selection are the three primary parts of the VM Migration Manager that work together to make the best migration decisions. In addition, the user mobility manager determines a user's likely cloudlets by combining data from the global mobility prediction manager with data from the user's local mobility management. At last, a group of jobs that need to be executed faster have their migration process started by the VM migration manager.

4. Meta-Heuristic ANT Colony-Based VM Migration Algorithm

We propose a method for migrating virtual machines (VMs) that involves transferring multiple VMs to multiple cloudlets simultaneously. Specifically, a cloudlet can reroute virtual machines to a group of cloudlets. In this scenario, the virtual machines act as bins, while the cloudlets serve as packs. Optimally packing these virtual machines into the fewest possible cloudlets minimizes overall job execution time while conserving resources. This problem is classified as NP-hard, indicating that no algorithm can guarantee an optimal solution in polynomial time. To address the VM migration challenge, we introduce a metaheuristic approach based on Ant Colony Optimization (ACO). This metaheuristic is employed because it is not problem-specific and can be adapted to various scenarios. In selecting the optimal virtual machine and cloudlet pair, we leverage ACO to navigate the dynamic cloud computing environment, as swarm optimization models are particularly effective in generating optimal solutions in such settings. ACO enhances the decision-making process for VM migration in distributed environments by accelerating computations. Cloudlets utilize data from mobile devices, the master cloud, and neighboring cloudlets to make informed decisions about VM migrations. Moreover, ACO's ability to learn and identify the characteristics of good solutions significantly increases the likelihood of finding the best solution while reducing the risk of poor decisions. The ACO metaheuristic algorithm mimics the behavior of virtual ants to construct a heuristic solution. We begin by creating a colony of ants, each of which applies a unique set of heuristics to solve smaller subproblems, eventually leading to a complete solution. Ants communicate through pheromones, refining their solutions as they go. Each ant develops its local optimal solution based on this pheromone trail and its heuristics. Ultimately, an optimal solution emerges when all ant colonies combine their best local solutions.

5. Cloudlet Selection: Pseudo-Random Proportional Rule

Ants select a cloudlet for each virtual machine according to a pseudo-random proportional rule (described below) when constructing the solution.

$$v = \begin{cases} \arg\max_{k \in V_{j,u}^{c}} \left([\tau_{u,k}]^{\alpha} \times [\eta_{u,k}]^{\beta} \right), & \text{if } q \leq q_0 \\ s, & \text{otherwise} \end{cases}$$

Local Pheromone Update

By selecting a VM pair to build a solution, an ant quickly updates the local pheromone value relative to the original pheromone value. Each ant uses the following relation to update its local pheromones:

$$\tau_{u,v}(t+1) = (1-\gamma_l) \times \tau_{u,v}(t) + \gamma_l \tau_0$$

Global Pheromone Update

Each pair of cloudlets and virtual machines has their pheromone values updated as soon as all the ants have constructed their local optimum solutions and updated the global optimal solution. The global pheromone values are updated using the following relation:

$$\tau_{u,v}(t+1) = (1-\gamma_g) \times \tau_{u,v}(t) + \gamma_g \Delta\tau_{u,v}$$

Algorithm 1: Cloudlet-Based Ant Colony-Based Virtual Machine Migration.

Steps:

Initialize:

Set the System tuning parameter α.
Configure the migration parameters γ_1 and γ_8.
Calculate the initial pheromone value γ_1.
Set the maximum iteration count MAX_IT.
Select Candidate VMs:
Use the pre-processing step to identify the candidate virtual machine set V^s.
Deploy Ants:
Begin adding ants to the ant colony A.
Iteration Loop:
While iteration ≤MAX_IT:

1. For Each Ant α∈ A:

Set k =0.
Repeat:
Select a VM in cloudlet j for VM u_k∈V^susing the pseudo-random proportional Formula (1).
Increment k= k + 1.
Until every VM is assigned to a cloudlet.

2. For Each VM k∈V^s :

Update the local pheromone using Formula (2).
Update the global pheromone using Formula (3).
Increment iteration = iteration+1.
Return:
The optimized VM-cloudlet pairs.

VM Migration Initiator

The best cloudlet for a virtual machine can be chosen using ant colony optimization. The PRIMIO method selects the currently active cloudlet as the target cloudlet after all virtual machines (VMs) have been assigned to it. The decision is made by calling the migration initiator for virtual machines. If the VM's destination and current cloudlets are different, the VM migration management will launch a virtual machine migration event. In every other case, the virtual machine is provided to the existing cloudlet without the need to initiate a VM migration event.

6. Results And Study

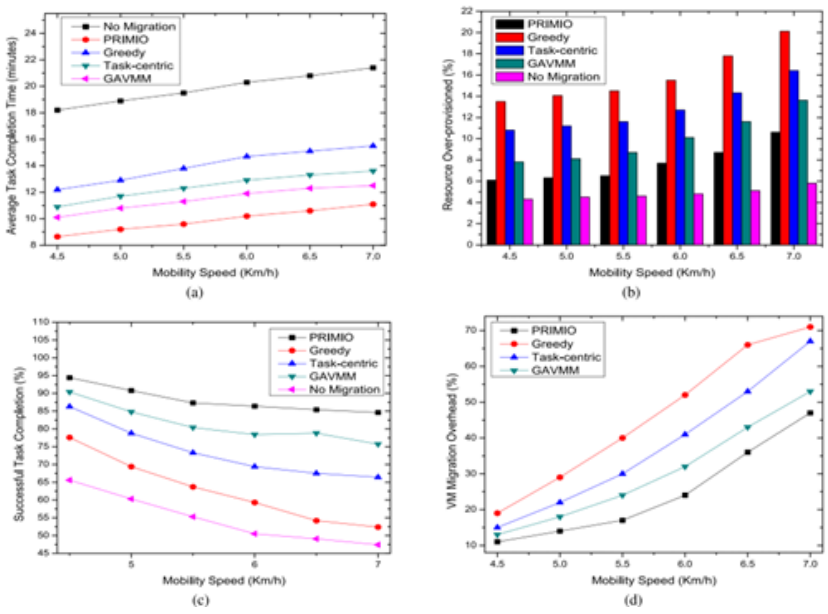

Figure 4. *Impacts of mobility speed. (a) Time required to complete a task on average. (b) Accumulated an excessive amount of resources. c) The work was completed successfully. (d) The load on virtual machines during migration.*

Since higher mobility speeds need more frequent migration of virtual machines, which in turn increases service downtime, as shown in Figure 4 (a), the average task lifetime grows as mobility speeds up. Additionally, there is an increase in the communication latency between the cloudlet and the user. Based on the graph, it can be shown that the No Migration policy results in longer average task execution times. The proportion of resources that are over-provisioned grows as the mobility speeds rise, as seen in Figure 4 (b). Tasks in the cloudlets migrate more frequently as user mobility speed increases, causing virtual machines (VMs) with more capacity than needed to be used. As the user's mobility speed increases, the percentage of tasks that are successfully executed within the deadline decreases, as shown in Figure 4 (c). One probable cause is that the task execution deadline is missed because the user's mobility causes an increase in communication delay with the virtual machine (VM) that is deployed in the cloudlet. This, in turn, causes the task lifetime to increase. This explains why fewer tasks are completed by the due date while using the No VM migration strategy. Increasing the speed of user mobility also increases the overhead of VM migration, as seen in Figure 4 (d). Since this method does not migrate any virtual machines (VMs), it does not create any migration overhead, which is why we will not be studying it. Although the VM migration overhead does increase with increasing user mobility speed, our suggested PRIMIO method claims that it does not significantly increase compared to existing ways. This is due to two primary causes.

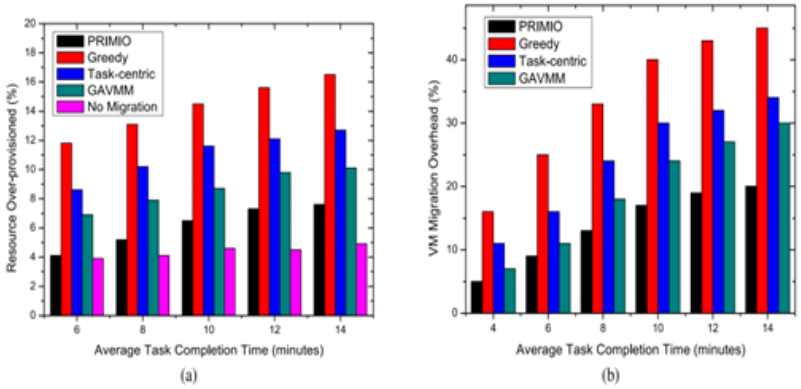

Figure 5. *Impacts of average job completion time. (a) Resources were allocated in excess. (b) Vagrant migration effort.*

The proportion of resources that are over-provisioned grows as the average lifetime of a job increases, as seen in Figure 5 (a). Nevertheless, PRIMIO's utilization of a shared VM migration approach guarantees appropriate utilization of system resources, preventing over-provisioned resources from seeing a significant increase. Unfortunately, overprovisioning of resources occurs rapidly with the greedy VM migration strategy due to its disregard for cloudlet resources in favor of user mobility. In contrast, virtual machine migration using a task-centric method is all-encompassing and wastes cloudlet resources. Finally, there's the No VM migration option, which doesn't increase the over-provisioned resources but has other negative effects. As the task execution lifespan in the cloudlets rises, the VM migration overhead develops, as seen in Figure 5 (b). The more time it takes to execute an action, the more work is involved in migrating virtual machines. When comparing the migration speed to the job lifespan, the system performance and the quality of service provided by user applications are both significantly impacted. Other approaches significantly increase VM migration overhead, as shown in this graph, when contrasted with our suggested PRIMIO VM migration solution. Additionally, at task execution lifetimes close to 10 minutes, the PRIMIO approach's virtual machine migration overhead hits a steady state. The reason behind this is that PRIMIO uses user mobility in conjunction with the joint-VM migration technique to start VM migration decisions.

7. Conclusion And Future Work

This work presents a paradigm for heterogeneous mobile cloud computing systems that takes mobility and resource awareness into account while migrating virtual machines. Improving the functionality of mobile health care applications in smart cities is the main objective. In this work, we tackle research issues related to shortening task-completion times and decreasing resource over-provisioning in healthcare applications that run on mobile cloud computing and are computationally and Big Data heavy. The suggested PRIMIO approach

starts virtual machine migration by taking cloudlet computational load and user mobility into account. With PRIMIO, users can take use of their mobility to choose the best solution, which means cloud resources are closer to them. During the entire task-execution time, PRIMIO reduces the total number of migrations by considering the load of the cloudlet to which the system intends to migrate the VM. In addition, we took resource over-provisioning rates into account while migrating virtual machines, which made the system make the most efficient use of its computing resources. We still have unfinished business in terms of optimizing task-computation time and data-access latency while taking fog clouds and crowd sourcing into account. With mobility and context awareness in mind, we will investigate new edge computing technologies to further optimize the time it takes to execute tasks.

References

[1]. Worldometers. World Population Forecast—Worldometers. 2019. Available online: https://www.worldometers.info/world-population/world-population-projections/ (accessed on 9 March 2021).

[2]. Ahvenniemi, H.; Huovila, A.; Pinto-Seppä, I.; Airaksinen, M. What are the differences between sustainable and smart cities? Cities 2017, 60, 234–245. [Google Scholar] [CrossRef]

[3]. United Nations. About the Sustainable Development Goals—United Nations Sustainable Development. Available online: https://sdgs.un.org/goals (accessed on 9 March 2021).

[4]. Cardullo, P.; Kitchin, R. Being a 'citizen' in the smart city: Up and down the scaffold of smart citizen participation in Dublin, Ireland. GeoJournal 2019, 84, 1–13. [Google Scholar] [CrossRef]

[5]. Desdemoustier, J.; Crutzen, N.; Giffinger, R. Municipalities' understanding of the Smart City concept: An exploratory analysis in Belgium. Technol. Forecast. Soc. Chang. 2019, 142, 129–141. [Google Scholar] [CrossRef]

[6]. Koubaa, A.; Aldawood, A.; Saeed, B.; Hadid, A.; Ahmed, M.; Saad, A.; Alkhouja, H.; Ammar, A.; Alkanhal, M. Smart palm: An IoT framework for red palm weevil early detection. Agronomy 2020, 10, 987

[7]. Rahman, A.; Hossain, M.S.; Alrajeh, N.A.; Guizani, N. B5G and Explainable Deep Learning Assisted Healthcare Vertical at the Edge: COVID-19 Perspective. IEEE Netw. 2020, 34, 98–105

[8]. Jin, X.B.; Yang, N.X.; Wang, X.Y.; Bai, Y.T.; Su, T.L.; Kong, J.L. Hybrid deep learning predictor for smart agriculture sensing based on empirical mode decomposition and gated recurrent unit group model. Sensors 2020, 20, 1334.

[9]. Lee, K.; Silva, B.N.; Han, K. Deep learning entrusted to fog nodes (DLEFN) based smart agriculture. Appl. Sci. 2020, 10, 1544. [Google Scholar] [CrossRef] [Green Version]

[10]. Guillén, M.A.; Llanes, A.; Imbernón, B.; Martínez, R.; Andrés, E.; Crespo, B.; Carlos, J. Performance evaluation of edge computing platforms for the prediction of low temperatures in agriculture using deep learning. J. Supercomput. 2021, 77, 818–840. [Google Scholar] [CrossRef]

[11]. Guillén-Navarro, M.A.; Martínez-España, R.; López, B.; Cecilia, J.M. A high-performance IoT solution to reduce frost damages in stone fruits. Concurr. Comput. Pract. Exp. 2021, 33, e5299. [Google Scholar] [CrossRef]

[12]. Rutqvist, D.; Kleyko, D.; Blomstedt, F. An Automated Machine Learning Approach for Smart Waste Management Systems. IEEE Trans. Ind. Informatics 2019. [Google Scholar] [CrossRef]

[13]. Hussain, A.; Draz, U.; Ali, T.; Tariq, S.; Irfan, M.; Glowacz, A.; Antonino Daviu, J.A.; Yasin, S.; Rahman, S. Waste Management and Prediction of Air Pollutants Using IoT and Machine Learning Approach. Energies 2020, 13, 3930.

[14]. Zhang, X.; Zhao, M.; Dong, R. Time-series prediction of environmental noise for urban iot based on long short-term memory recurrent neural network. Appl. Sci. 2020, 10, 1144. [Google Scholar] [CrossRef] [Green Version]

[15]. Han, T.; Muhammad, K.; Hussain, T.; Lloret, J.; Baik, S.W. An Efficient Deep Learning Framework for Intelligent Energy Management in IoT Networks. IEEE Internet Things J. 2020. [Google Scholar] [CrossRef]

[16]. Liu, X.; Xiao, Z.; Zhu, R.; Wang, J.; Liu, L.; Ma, M. Edge sensing data-imaging conversion scheme of load forecasting in smart grid. Sustain. Cities Soc. 2020, 62, 102363. [Google Scholar] [CrossRef]
[17]. Taïk, A.; Cherkaoui, S. Electrical load forecasting using edge computing and federated learning. In Proceedings of the ICC 2020-2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6
[18]. 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.
[19]. Awais, M.; Raza, M.; Singh, N.; Bashir, K.; Manzoor, U.; ul Islam, S.; Rodrigues, J.J. LSTM based Emotion Detection using Physiological Signals: IoT framework for Healthcare and Distance Learning in COVID-19. IEEE Internet Things J. 2020.
[20]. Garmaroodi, M.S.S.; Farivar, F.; Haghighi, M.S.; Shoorehdeli, M.A.; Jolfaei, A. Detection of Anomalies in Industrial IoT Systems by Data Mining: Study of CHRIST Osmotron Water Purification System. IEEE Internet Things J. 2020, 4662, 1