mathematics

MDPI

Article

Optimized Task Group Aggregation-Based Overflow Handling on Fog Computing Environment Using Neural Computing

Harwant Singh Arri [1], Ramandeep Singh [1], Sudan Jha [2], Deepak Prashar [1], Gyanendra Prasad Joshi [3,*] and Ill Chul Doo [4,*]

[1] School of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab 144411, India; hs.arri@lpu.co.in (H.S.A.); ramandeep.singh@lpu.co.in (R.S.); deepak.prashar@lpu.co.in (D.P.)
[2] School of Computational Sciences, CHRIST (Deemed to be University), Mariam Nagar, Meerut Road, Delhi NCR, Ghaziabad 201003, India; jhasudan@ieee.org
[3] Department of Computer Science and Engineering, Sejong University, Gwangjin-gu, Seoul 05006, Korea
[4] Artificial Intelligence Education, Hankuk University of Foreign Studies, Dongdaemun-gu, Seoul 02450, Korea
* Correspondence: joshi@sejong.ac.kr (G.P.J.); dic@hufs.ac.kr (I.C.D.)

Abstract: It is a non-deterministic challenge on a fog computing network to schedule resources or jobs in a manner that increases device efficacy and throughput, diminishes reply period, and maintains the system well-adjusted. Using Machine Learning as a component of neural computing, we developed an improved Task Group Aggregation (TGA) overflow handling system for fog computing environments. As a result of TGA usage in conjunction with an Artificial Neural Network (ANN), we may assess the model's QoS characteristics to detect an overloaded server and then move the model's data to virtual machines (VMs). Overloaded and underloaded virtual machines will be balanced according to parameters, such as CPU, memory, and bandwidth to control fog computing overflow concerns with the help of ANN and the machine learning concept. Additionally, the Artificial Bee Colony (ABC) algorithm, which is a neural computing system, is employed as an optimization technique to separate the services and users depending on their individual qualities. The response time and success rate were both enhanced using the newly proposed optimized ANN-based TGA algorithm. Compared to the present work's minimal reaction time, the total improvement in average success rate is about 3.6189 percent, and Resource Scheduling Efficiency has improved by 3.9832 percent. In terms of virtual machine efficiency for resource scheduling, average success rate, average task completion success rate, and virtual machine response time are improved. The proposed TGA-based overflow handling on a fog computing domain enhances response time compared to the current approaches. Fog computing, for example, demonstrates how artificial intelligence-based systems can be made more efficient.

Keywords: fog computing; resource scheduling; overflow handling; virtual machines; TGA; ABC; neural computing and ANN

Citation: Arri, H.S.; Singh, R.; Jha, S.; Prashar, D.; Joshi, G.P.; Doo, I.C. Optimized Task Group Aggregation-Based Overflow Handling on Fog Computing Environment Using Neural Computing. *Mathematics* **2021**, *9*, 2522. https://doi.org/10.3390/math9192522

Academic Editor: Vladimir M. Vishnevsky

Received: 21 August 2021
Accepted: 2 October 2021
Published: 7 October 2021

1. Introduction

Over the past few years, the amount of information generated worldwide has increased dramatically, necessitating enormous storage facilities [1]. In network architecture, the word cloud or fog refers to how network engineers locate and connect to various network devices to access or store data [2]. There are two terms commonly used to describe fog computing: "fogging" and "fog networking." To put it another way, fog is a distributed computing environment where data and applications are spread out among multiple data sources and the cloud. Basic analytic services will be brought to the network edge, lowering the amount of data that need to be transmitted over the network and increasing overall network efficiency and performance as a result of a better location. Security can also benefit from fog computing, as it can split bandwidth flow and add additional firewalls to

a network for extra safety. This network's design resembles a cloud; therefore, the name "Cloud". Figure 1 shows the cloud computing architecture in broad terms.

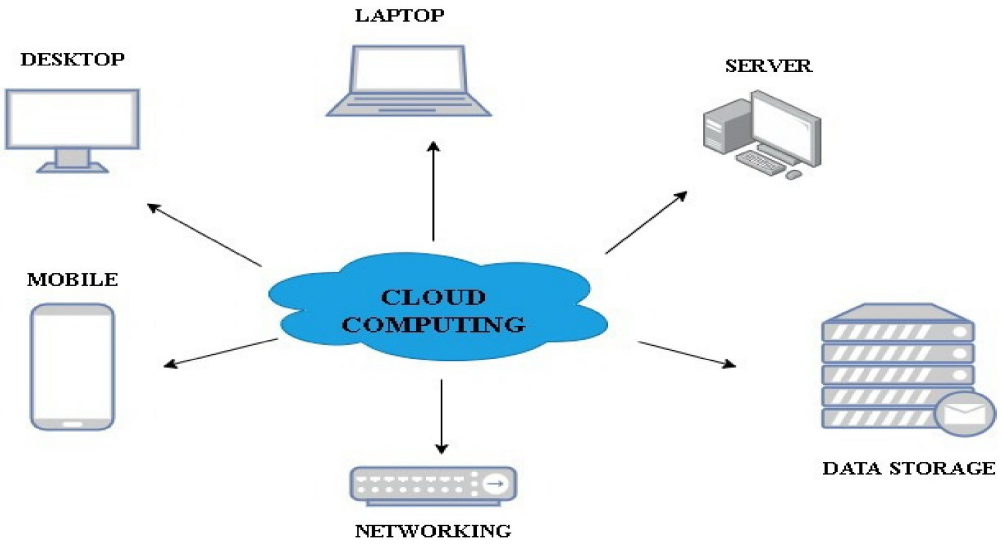

Figure 1. General cloud model representation.

Cloud computing, sometimes known as fog computing, is a relatively new concept that uses the Internet and remote servers to store data and run applications. Instead of being directly tied to a server, cloud computing delivers IT services via the Internet using web-based tools and apps. One can store files in a remote database rather than on a hard disc or another local storage medium, as shown in Figure 2, which illustrates cloud-based storage.

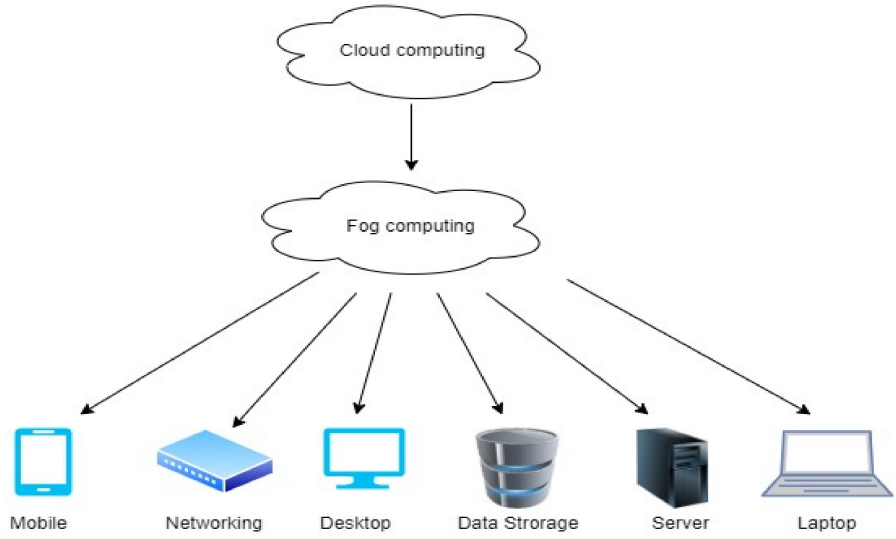

Figure 2. Fog computing platform representation.

Fog computing is a revolutionary new approach to delivering IT services. Many business owners hope to use this method to consolidate all of their IT services into one place. Such services have grown in popularity as the information society has progressed, as entrepreneurs assist integrated companies in solving IT challenges swiftly and efficiently. For the purposes of this discussion, let us say the fog computing environment is about the needs of IT firms. Cloud users can access a wide range of services via the Internet and

in real-time. Using cloud computing as an example, let us categorize the most prevalent services [3–5].

An application, some storage, and a network are all part of the fog computing system. For enterprises and individuals around the world, each department caters to distinct needs with specialized products. The front and rear ends of fog computing include two components. The front end of a cloud computing system serves the needs of the end-user. A cloud computing platform's access interfaces and applications make up this infrastructure. A cloud's back end includes the resources needed to provide cloud computing services. A cloud service provider manages this virtual computer, server, data storage, and security mechanism, among other things [6]. Figure 3 shows the architecture of the fog computing concept.

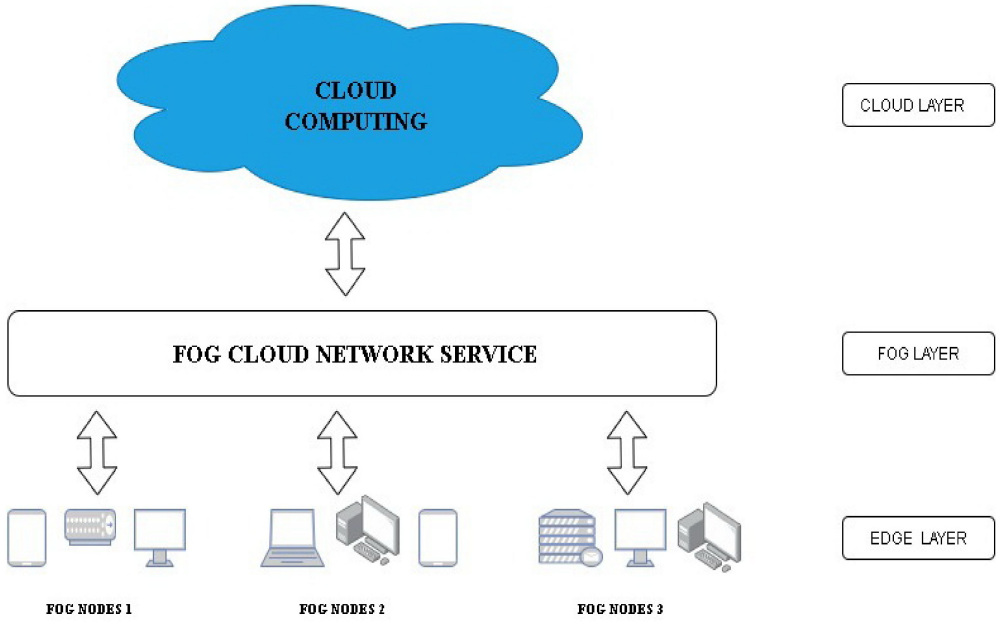

Figure 3. The architecture of the fog computing model.

In addition, with the evidence of neural computing as one of the pioneers in the current era, utilization of the same in various applications such as fog and edge computing has become necessary, and also it makes the applications optimized. Neural computing has become a widespread choice owing to its adaptability and better results pertaining to the applications on which it is applied. Hence, in this work, some advanced forms of techniques of neural computing are utilized.

ANN as machine learning combined with artificial Bee Colony (ABC) and Task Group Aggregation (TGA) methods [7] are used in Figure 3 to show the suggested fog computing model's architecture. The existing fog computing model faces some crucial issues:

Overflow Handling: This approach allows the servers or virtual machines to maintain data flow in fog computing. It helps to handle the overflow problem based on the Quality of Service (QoS) parameters, such as Resource Scheduling Efficiency, Energy uses and Response Time, and Average Success Rate.

Load Balancing: In order to maximize time efficiency and make optimal use of resources, it distributes jobs or loads among multiple system nodes. By collecting and sorting the expected execution time from each of the produced virtual machines, the load balancing process begins by classifying the number of tasks included within each VM. Rearranging the classified jobs in each VM according to the most recent execution period is how load balancing is accomplished.

Virtual Machine Migration: In order to help data center operators meet performance targets, boost resource efficiency and connectivity locality, reduce performance hotspots,

and achieve fault tolerance and reduce energy consumption, virtual machine (VM) migration is an effective management strategy that allows data center operators to adjust the location of virtual machines (VMs). Because it emulates the underlying hardware, a virtual machine has an interface that is identical to that of the physical machine. Virtualization's benefits include better resource utilization, a higher level of abstraction, and more scalable and adaptable architecture.

Fault Tolerance: It is possible to run virtual computers with fault tolerance, even if part of the device fails. This method moves the virtual computer from a single actual server to another based on an estimate of collapse. While preserving the device performance, the fault-tolerant migration method makes the physical server more available. This research has been motivated by a number of factors, with the principal one being the development of an overflow handling model that combines the ANNABC and TGA algorithms. The key influences in this investigation are enumerated as follows:

1. To address the overflow problem on fog servers or virtual machines, we provide an ANN-oriented overflow management model with a TGA method for a fog computing environment;
2. TGA with ABC is used as a classifier to detect overflow problems in ANN;
3. The suggested fog computing overflow control model is tested by comparison to the current state-of-the-art virtual machine efficiency for resource scheduling average success rate, average task completion success rate, and virtual machine response time.

The rest of the article follows this structure. Section 2 presents an examination of the connected works. Methods are described in Section 3, and results are shown in Section 4. In the end, Section 5 summarized the findings and forecasts for cloud computing's virtual machine load optimization in the coming years.

2. Related Work

The essential features and challenges in the cloud or fog computing model are uncovered by reviewing a variety of previous works. Cloud service provisioning resource scheduling approach with load balancing was presented by V. Priya et al. in 2018 [8]. Researchers coupled integrated resource scheduling with a load balancing algorithm in this work to provide efficient cloud service provisioning. A fuzzy-based multi-dimensional resource scheduling (FRMS) model based on the MQLO with the MRSQN method was used to improve virtual machine resource scheduling efficiency in the cloud. The MQLO algorithm was then used to dynamically pick a request from a class, increasing Virtual Machine utilization through efficient and equal load balancing. A load adjusting equation was used to avoid asset under- and overutilization, reducing idle time for each type of solicitation. The proposed method improved execution in terms of typical rate of success, asset planning expertise, and turnaround time, and per the findings of recreations aimed at evaluating the sufficiency in a cloud server. Compared to the best-in-class works, leisure research showed that the technique increases asset booking effectiveness by 7% and decreases response time by 35.5 percent.

In order to reduce energy consumption and completion time, Guo et al. [9] developed an energy effectual vigorous offloading and device scheduling strategy. The primary goal of this study was to find a solution to the issue of excessive energy usage when faced with strict limits. A significant reduction in energy costs could be achieved with sufficient task dependence and completion time. Clock frequency control and broadcast power allocation were employed by the author a lot, along with a technique for selecting computation offloading. A task's workload determines which computation method to use, and doing so increases completion time. Mukherjee et al. [10] proposed handling the problem of mobile device mobility, as these devices move at different speeds, and (ii) task allocation becomes difficult if the mobile device's position changes. The primary challenge was handled by utilizing a technique that utilized remote cloud servers to carry out the operation on mobile devices. If the connection is lost, the server sends a message (pushing alert) to restore the server and mobile node. To address the issue of location, the VM transfer concept was

applied. The sample must be transferred from the prior download platform to the new download platform to continue unloading. According to the results of the analysis, the suggested work cuts power consumption by 30 to 63%. When using a mobile device in and out of the university building, different speeds produce varied consequences.

The m-cloud system's suggested malfunction detection and recovery policies have been found [11]. The algorithm's performance has been studied through a series of tests. Studies have shown how an integrated decision strategy and a planned system can help users make decisions about which mobile devices to use and which cloud resources to use. SLA, energy usage, and makespan have all been tallied. Resource scheduling was made better utilizing optimization approaches, but when it comes to loading balancing, there are some concerns connected to the classification of overloaded or underloaded cloud servers, which must be addressed [12]. Researchers [13–19] have done similar work on fog computing load balancing and the use of neural computing ideas. The use of neural computing concepts and methods in the fog environment has also lately become popular among scientists [20–25].

To control overflow concerns in fog computing, we employed an artificial neural network (ANN) as a classifier to recognize overloaded or underloaded servers or virtual machines (VMs) and balance them based on their basic metrics resource scheduling average success rate, average task completion success rate and virtual machine response time. The proposed model is a hybridization of artificial neural network (ANN) and Task Group Aggregation (TGA) algorithm in fog computing, which increases the reaction time as well as the success rate. This model is an energy optimization-based tradeoff scheme in a fog computing environment. Moreover, the VMs are allotted to the ABC algorithm, which delivers the optimized list of VMs. At last, ANN as a supervised machine learning method is used to distinguish between faulty and normal VMs to avoid the failure rate and reduction in job completion time [25].

3. Method of Model

Using the overflow handling idea and ANN as machine learning in conjunction with ABC and TGA algorithms for fog computing, we provide the methodology and algorithms of the suggested model for fog computing in this section of the research article. The following are the model's procedural phases in a fog computing environment.

Proposed Simulator: To begin, we create MATLAB 2016a software simulations for the suggested fog computing environment model using the graphical user interface paradigm.

Figure 4 illustrates the used simulation area of the proposed scenario in the MATLAB software. In the simulator, the x-axis represents the width of the network (2000 m), and the y-axis represents the height of the network (2000 m). So, a total area of the simulator of 2000×2000 m^2 is used for deployment.

User and Server Deployment in the Simulator:

When the simulation is completed, the users and servers are placed in the region depicted in Figure 5. A few basic parameters were provided to each server, user, and VM for simulation purposes after the deployment was complete. These parameters were then sorted in descending order to identify the most efficient server with the greatest amount of CPU, Memory, and Bandwidth available. First unsorted data will be sorted, and then, after execution for a load balancing and resource scheduling, the data will be used in the simulation model. We used TGA, ABC, and ANN algorithms to calculate the model's performance measures, such as Mean Rate of success, Resource Provisioning Effectiveness, Power Consumption, and Turnaround Time are all important factors to consider.

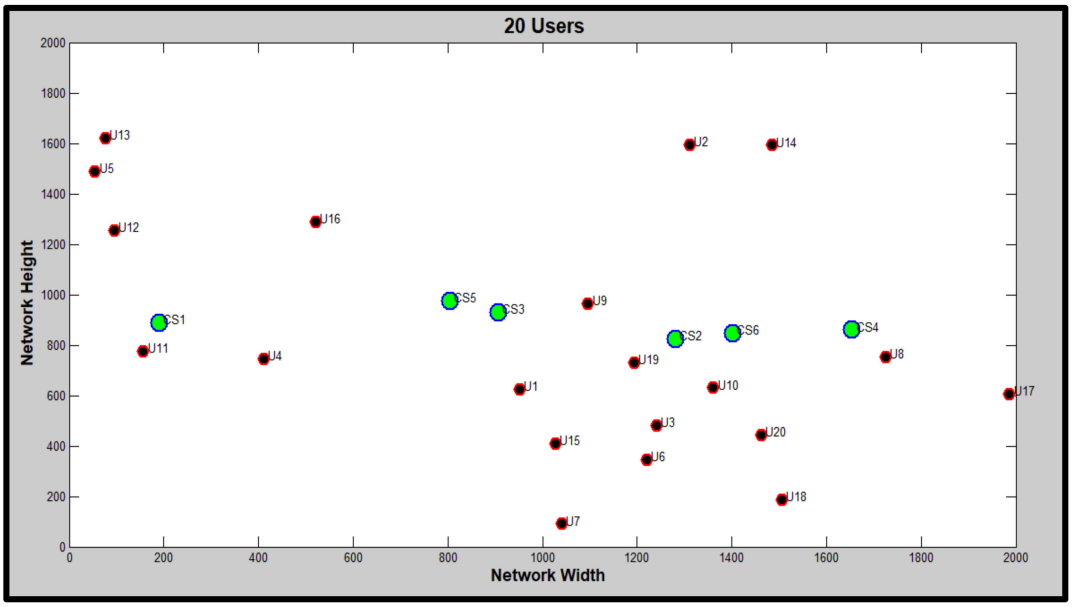

Figure 4. The architecture of the Fog Computing Model.

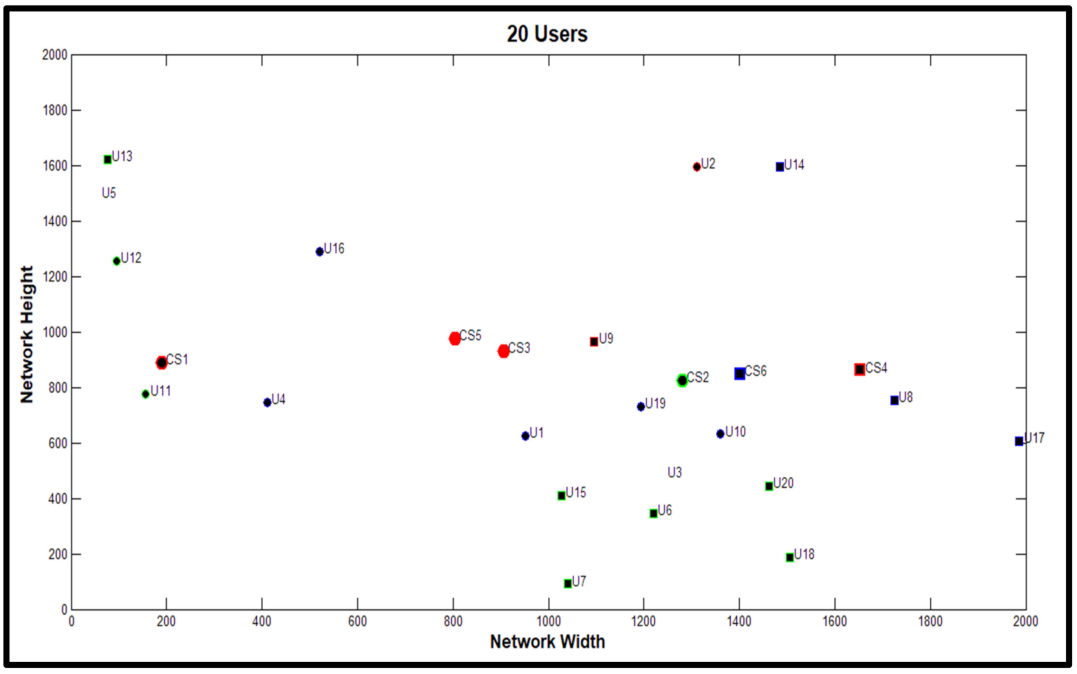

Figure 5. Resource Allocation.

The mentioned Algorithm 1 of TGA is responsible for the task group aggregation in the proposed work, and then, the concept of ABC is used to optimize the resource allocation based on their fitness using the given Algorithm 2:

Algorithm 1. Algorithm of TGA (Task Group Aggregation)

Input: U→Number users
C→Number of servers
Output: AS→List of Allocated Server

Start
Define RTF = [sCPUsMemsBW]//Assign the RTF (Resource Threshold Factor) using the basic parameters of servers like their CPU, RAM, and bandwidth
For each U
TimInt = random//Requests at Time Interval
DemR= max([uCPUuMemuBW])//user demand for resources
UtilRate= DemR×TimInt//Resource Utilization Rate
If max (UtilRate)<=max (RTF)
AvgPTimeSer(i)=DemR(i)/(1-
UtilRate(i))AvgPTimeAllSer(i)=(AvgPTimeSer(i)×TimInt(i))/(TimInt(i))
AS-LIST =ceil(cServer × rand)
End—If
End—For
Return: AS-List as a list of allocated servers
End—Algorithm

Algorithm 2. Algorithm of ABC (Artificial Bee Colony) with Fitness Function

Input: AS→ List of Allocated Server
f (fit)→Fitness Function of ABC
Output: OAS→Optimized List of Allocated Server

Start
Initialize ABC algorithm with operators and parameter—Iterations (ITR)
– Bee Size (S)
– Lower Bound (LB)>
– Upper Bound (UB)
– Number of Variables (Nvar)
Calculate Size of AS, SZ = Size (AS)
Define Fitness function, f (fit)

$$\mathrm{f}\,(\mathrm{fit}) = \begin{cases} \text{True; if condition fullfill for allocation} \\ \text{False; otherwise} \end{cases}$$

For each ITR & SZ
$\mathrm{E_{Bee}} = \sum_{i=1}^{S} \mathrm{Power}(\mathrm{VMs}(\mathrm{AS}))$//Select one by on VMs from allocated serves list
$\mathrm{O_{Bee}} = \frac{\sum_{i=1}^{S} \mathrm{Power}(\mathrm{VMs}(\mathrm{AS}))}{S_Z}$//Threshold
f(fit) = fitness function//which is define above

$$\mathrm{Cserver_{prop}} = \mathrm{ABC}(\mathrm{S}, \mathrm{LB}, \mathrm{UB}, \mathrm{M}, \mathrm{Nvar}, \mathrm{f}(\mathrm{fit}))$$

End—For
For each CserverProp
OAS = Count (find (AS == CserverProp))
End—For
Return: OAS→Optimized List of Allocated Server
End—Algorithm

When a list of optimized allocation of servers is returned by Algorithm 2 of ABC based on their fitness, the machine learning idea of ANN is used to deal with overflow problems in order to identify overburdened or underburdened servers and VMs and balance them on the basis of their basic properties.

As a next step, we look at resource scheduling, energy consumption, and reaction time to determine the suggested overflow handling concept's performance characteristics as given in Algorithm 3. This is done using the combination of ANN machine learning with ABC and TGA algorithms for fog computing environments. The proposed approach

using task group aggregation and optimized ANN [21] includes overflow handling as a critical component. These three algorithms must be tuned in a fog computing environment to achieve optimal task group aggregation based on an overflow management mechanism. The architecture of ANN is shown in Figure 6 with a description, and scalable traffic management has recently been created in fog computing for data centers to balance traffic load and service quality.

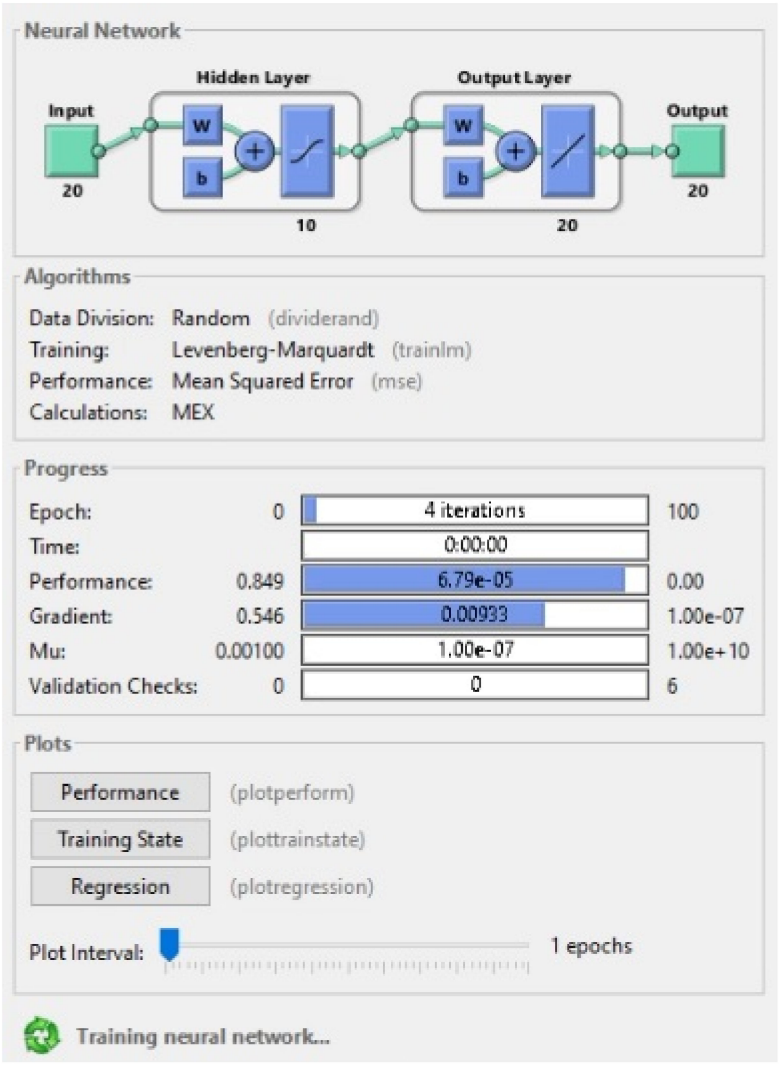

Figure 6. ANN Architecture proposed work.

Algorithm 3. Algorithm of ANN for Overflow Handling (Artificial Neural Network) with TGA

Input: OAS→Optimized List of Allocated Server
T-Data→VM properties as training data
TR→No. of VMs as a target in OAS
N→Neurons
Output: VAS→ValidList of Allocated Server

Start
For each cServer
If OAS (individual)>Average (OAS)
T-Data= [sCPUsMemsBW]
Targetc []
End—If
End—For
Foreach T-Data
IfsCPU(Individual)>= Average(sCPU)
Target (1)= 1
Else ifsCPU(Individual)<minimum (sCPU))
Target (2)= 1
Else
Target (3) =1
End—If
End—For
Call and set the ANN
Set, Fog-Net = Newff (T-Data, TR, N)
Fog-NetEpoch = 1000
Fog-Net Training Data Ratio = 70%
Fog-Net Testing Data Ratio = 15%
Fog-NetValidation Data Ratio = 15%
Fog-Net = Train (Fog-Net, T-Data, TR)
Properties Current VM in OAS = VMC
VM Characteristics = simulate (Fog-Net, VMC)
If VM Characteristics is valid and not overloaded
VAS = Validated
Else
VAS = Maybe under or overloaded
End
Return: VAS as the list of valid allocated server
End—Algorithm

Input, hidden, and output layers are all components of an ANN's architecture. Users, servers, and VM attributes are used to train the ANN for scheduling and balancing. Input parameters include things such as the amount of energy used by the server to execute a task and the amount of CPU time used in conjunction with bandwidth utilization on the server or Virtual Machine. Any errors generated are transmitted back to the hidden layer so that nodes' characteristics can be tweaked until the desired result is achieved. So, the network may be taught with the least amount of error possible. There are N interconnected neurons in the network, as shown in Figure 7 by the arrow. These neurons independently update each neuron's activation function. The Mean Square Error (MSE) is the error caused by the ANN network during training.

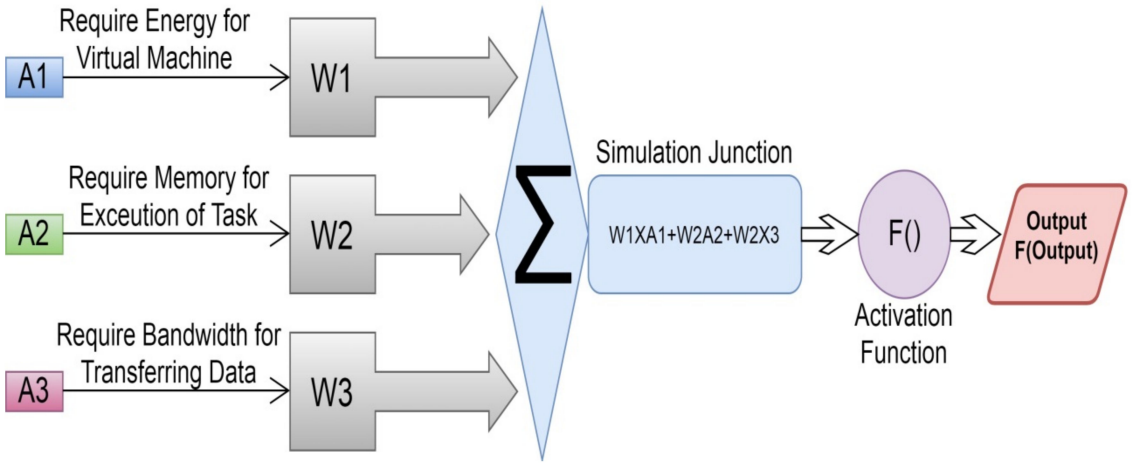

Figure 7. ANN Architecture.

Using ANN as machine learning with ABC and TGA method for a fog computing environment, we offer an experimental section after the formation of the model. The model includes simulation results of the proposed fog computing model based on the overflow handling idea and compares it to the previous work.

4. Results and Discussion

Here, we discourse the simulation consequences of the suggested fog computing model, which is based on the overflow handling idea and uses ANN as a machine learning algorithm together with the ABC and TGA algorithms for fog computing. First and foremost, we will go over the findings of the study's computer simulations.

As shown in Figure 8, the suggested model has an average success rate based on user requests ranging from five to one hundred. According to our findings, the average success rate decreases as the number of requests from the user rises.

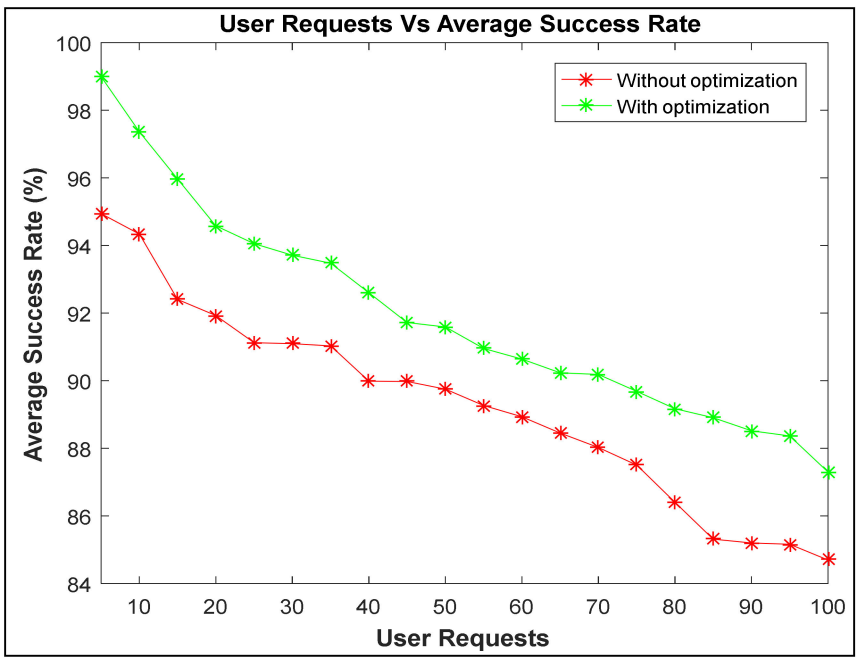

Figure 8. Average Success Rate.

Figure 9 depicts resource scheduling efficiency of user requests, taking into account a range of customer requests from 5 to 100. Based on our observations, resource scheduling efficiency ranges from 80 to 96 percent as the number of requests increases.

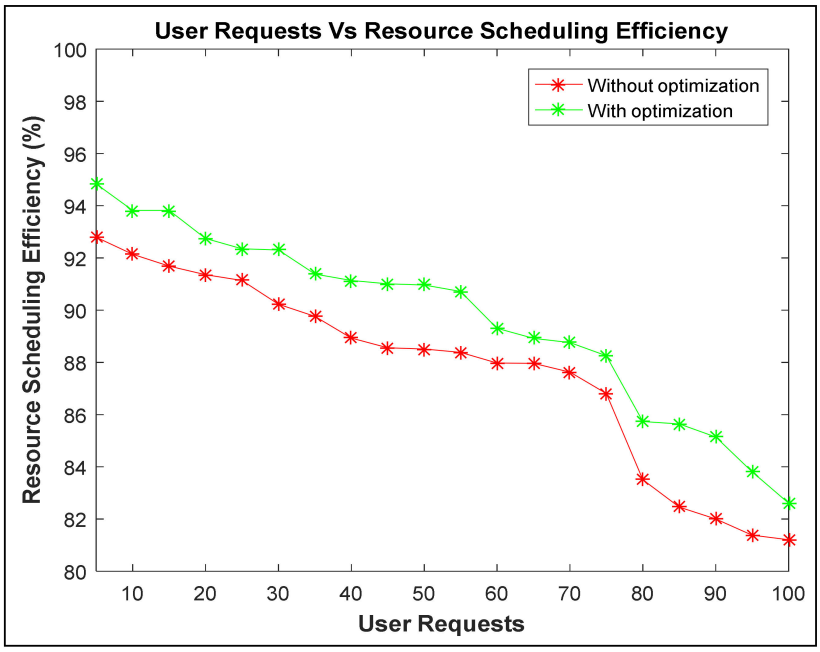

Figure 9. Resource Scheduling Efficiency.

As shown in Figure 10, the energy usage varies from 1 to 100 users when considering user demands. As a result of our research, we have discovered that energy usage is rising along with user demand. Responding to user requests in a reasonable amount of time is shown in Figure 11, which takes into account queries from between five and one hundred users. We have noticed an increase in response time and an increase in user requests as a result of our observations.

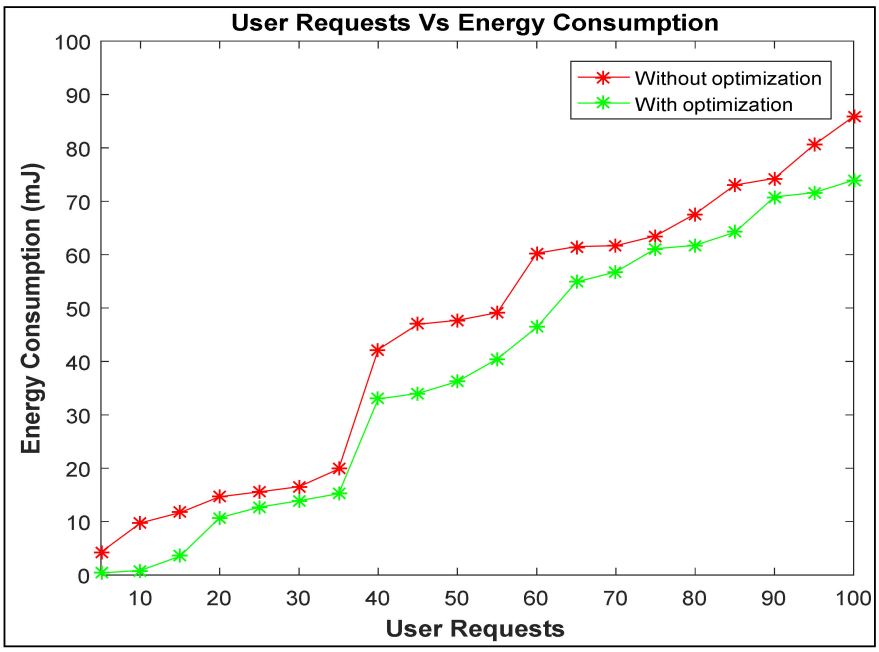

Figure 10. Energy Consumption.

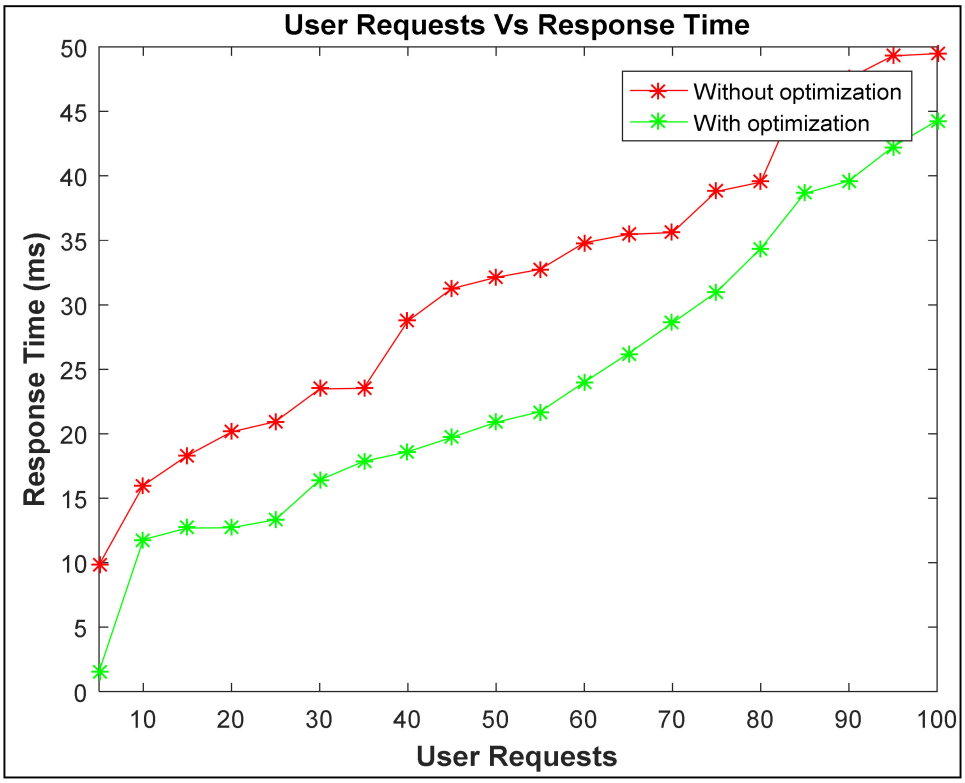

Figure 11. Response Time.

We compare the resulting parameters with and without optimization for all parameters so that the proposed model's efficacy can be easily determined. Table 1 compares our simulation results with those of V. Priya et al. [8] once the experiments are complete.

Table 1. Comparison of Parameters with Advantages and Disadvantages.

Parameters	Current/Existing Model [26]	Implemented/Proposed Model	Advantage and Disadvantage of Proposed Model
Virtual Machine Efficiency For Resource Scheduling	In the existing models, average of resource scheduling is 93.10%.	In the proposed model, average of resource scheduling is 97.20%.	The success rate of resource scheduling is increased because more new tasks will be allocated to the virtual machines as per energy consumption. However, this work can be extended by considering the concept of clustering mechanism with optimization technique to find out the over- and underloaded VMs so that we can manage the task allocation properly.
Average Task Completion Success Rate	In the existing models, the average success rate is 94.20.	The proposed models' average success rate is 98.10%.	The proposed scheme achieved a far better result than the other existing work based on the job completion time. It mitigates the job scheduling problem in the fog computing environment. Additionally, it gratifies the service requests of operators based on the optimal tradeoff scheme. However, the energy consumption is still massive for a cost-effective prototype.
Virtual Machine Response time	In the existing model, average response time is 5.65 ms.	In the proposed model, average response time is 4.55 ms.	The issue of resource scheduling and task overflow handling in fog computing can be handled by reducing response time and balancing the load on servers. This provides the optimal tradeoff scheme between user and operator.

Table 1 and Figure 12 show that using the deep learning-based ANN classifier improves the projected work efficiency over the prior work.

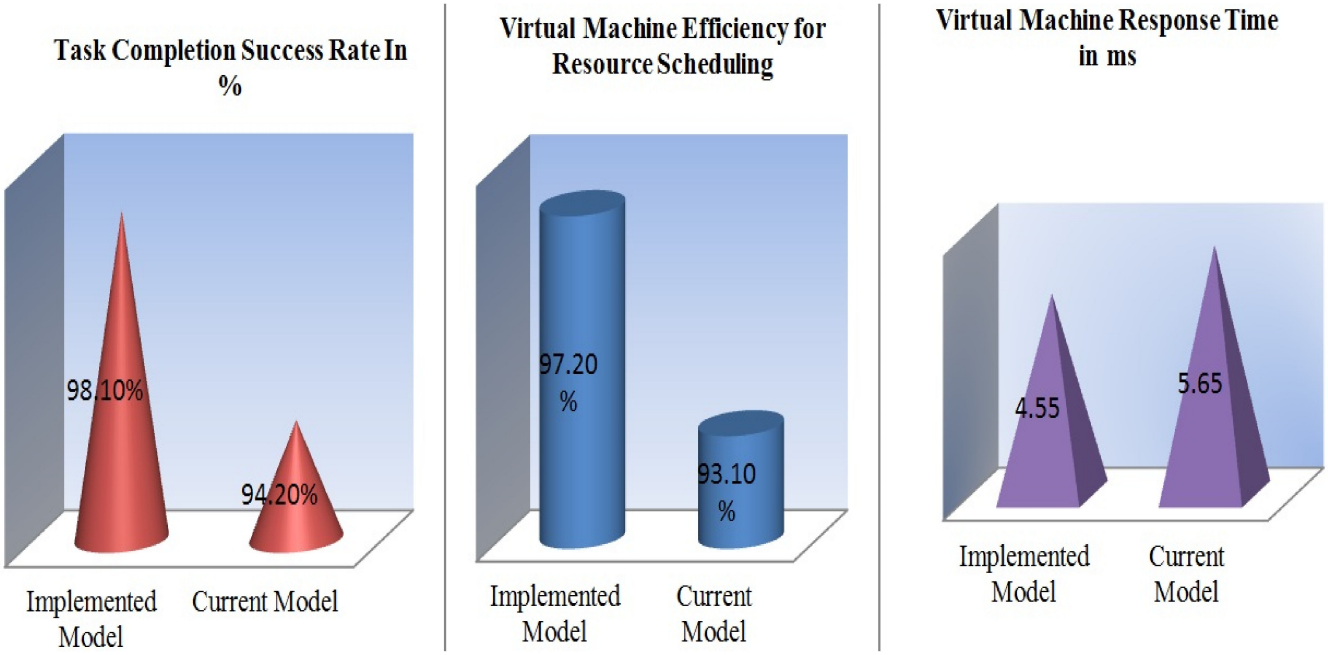

Figure 12. Comparison of Parameters.

5. Conclusions

This paper suggested a fog computing environment where ABC and TGA algorithms are combined with an ANN as machine learning for overflow handling and work schedule. In fog computing, we looked at the problem of resource scheduling and overflow handling to see how we could reduce the model's response time while also using less energy and balancing the load on the servers. A major problem is managing the arrival of new users' tasks while also balancing the server's load. As a consequence, we used the ANN concept to detect overloaded or underloaded servers and transfer the load to the VMs operating on servers were feasible for the load task optimization. Although the proposed model has superior virtual machine efficiency for resource scheduling, average success rate, and average task completion success rate compared to the existing work, it still consumes a significant amount of energy despite being cost-effective. An improved ANN will be used in the cloud's future to produce better intelligible models by applying swarms-based metaheuristic algorithms built on top of the idea of optimized artificial neural networks (ANNs).

Author Contributions: Conceptualization, H.S.A.; data curation, H.S.A.; formal analysis, R.S.; funding acquisition, G.P.J. and I.C.D.; methodology, S.J. and D.P.; project administration, G.P.J. and I.C.D.; resources, G.P.J.; supervision, S.J., G.P.J. and I.C.D.; visualization, D.P.; writing—original draft, H.S.A. and R.S.; writing—review and editing, G.P.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Hankuk University of Foreign Studies Research Fund of 2021. This research was also supported by the MIST (Ministry of Science, ICT), Korea, under the National Program for Excellence in SW), supervised by the IITP (Institute of Information & communications Technology Planning & Evaluation) in 2021 (2019-0-01816).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not Applicable.

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

References

1. Zhao, J.; Yang, K.; Wei, X.; Ding, Y.; Hu, L.; Xu, G. A heuristic clustering-based task deployment approach for load balancing using Bayes theorem in cloud environment. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *27*, 305–316. [CrossRef]
2. El Amrani, C.; Gibet Tani, H. Smarter round robin scheduling algorithm for cloud computing and big data. *J. Data Min. Digit. Humanit.* **2018**, *2016*, 1–8.
3. Porkodi, V.; Singh, A.R.; Sait, A.R.W.; Shankar, K.; Yang, E.; Seo, C.; Joshi, G.P. Resource Provisioning for Cyber–Physical–Social System in Cloud-Fog-Edge Computing Using Optimal Flower Pollination Algorithm. *IEEE Access* **2020**, *8*, 105311–105319. [CrossRef]
4. Rahmani, A.M.; Gia, T.N.; Negash, B.; Anzanpour, A.; Azimi, I.; Jiang, M.; Liljeberg, P. Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Gener. Comput. Syst.* **2018**, *78*, 641–658. [CrossRef]
5. Verma, D.; Rana, S. Deep learning based load balancing using multidimensional queuing load optimization algorithm for cloud environment. *Int. J. Eng. Sci. Res. Technol.* **2020**, *9*, 156–167.
6. Ashouraei, M.; Khezr, S.N.; Benlamri, R.; Navimipour, N.J. A New SLA-Aware Load Balancing Method in the Cloud Using an Improved Parallel Task Scheduling Algorithm. In Proceedings of the 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), Barcelona, Spain, 6–8 August 2018; pp. 71–76.
7. Belgaum, M.R.; Soomro, S.; Alansari, Z.; Alam, M. Cloud service ranking using checkpoint-based load balancing in real-time scheduling of cloud computing. In *Progress in Advanced Computing and Intelligent Engineering*; Springer: Singapore, 2018; pp. 667–676.
8. Priya, V.; Kumar, C.S.; Kannan, R. Resource scheduling algorithm with load balancing for cloud service provisioning. *Appl. Soft Comput.* **2019**, *76*, 416–424. [CrossRef]
9. Guo, S.; Liu, J.; Yang, Y.; Xiao, B.; Li, Z. Energy-Efficient Dynamic Computation Offloading and Cooperative Task Scheduling in Mobile Cloud Computing. *IEEE Trans. Mob. Comput.* **2019**, *18*, 319–333. [CrossRef]
10. Mukherjee, A.; Roy, D.G.; De, D. Mobility-aware task delegation model in mobile cloud computing. *J. Supercomput.* **2019**, *75*, 314–339. [CrossRef]
11. Zhou, B.; Dastjerdi, A.V.; Calheiros, R.; Srirama, S.N.; Buyya, R. mCloud: A Context-Aware Offloading Framework for Heterogeneous Mobile Cloud. *IEEE Trans. Serv. Comput.* **2017**, *10*, 797–810. [CrossRef]
12. Hung, T.C.; Hieu, L.N.; Hy, P.T.; Phi, N.X. MMSIA: Improved max-min scheduling algorithm for load balancing on cloud computing. In Proceedings of the 3rd International Conference on Machine Learning and Soft Computing, Dalat, Vietnam, 25–28 January 2019; pp. 60–64.
13. Mishra, S.K.; Sahoo, B.; Parida, P.P. Load balancing in cloud computing: A big picture. *J. King Saud Univ. Comput. Inf. Sci.* **2020**, *32*, 149–158. [CrossRef]
14. Atlam, H.F.; Walters, R.J.; Wills, G.B. Fog Computing and the Internet of Things: A Review. *Big Data Cogn. Comput.* **2018**, *2*, 10. [CrossRef]
15. Subhulakshmi, R.; Suryagandhi, S.; Mathubala, R.; Sumathi, P. An evaluation on Cloud Computing Research Challenges and Its Novel Tools. *Int. J. Adv. Res. Basic Eng. Sci. Technol.* **2016**, *2*, 69–76.
16. Rashid, A.; Chaturvedi, A. Cloud computing characteristics and services: A brief review. *Int. J. Comput. Sci. Eng.* **2019**, *7*, 421–426. [CrossRef]
17. Kumar, P.; Kumar, R.; Gupta, G.P.; Tripathi, R. A Distributed framework for detecting DDoS attacks in smart contract-based Blockchain-IoT Systems by leveraging Fog computing. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, 4112. [CrossRef]
18. Kumar, P.; Gupta, G.P.; Tripathi, R. An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for IoMT networks. *Comput. Commun.* **2021**, *166*, 110–124. [CrossRef]
19. Mutlag, A.A.; Ghani, M.K.A.; Arunkumar, N.; Mohammed, M.A.; Mohd, O. Enabling technologies for fog computing in healthcare IoT systems. *Futur. Gener. Comput. Syst.* **2019**, *90*, 62–78. [CrossRef]
20. Bonomi, F.; Milito, R.; Natarajan, P.; Zhu, J. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*; Springer: Cham, Switzerland, 2014; pp. 169–186.
21. Kumar, D.P.; Amgoth, T.; Annavarapu, C.S.R. Machine learning algorithms for wireless sensor networks: A survey. *Inf. Fusion* **2019**, *49*, 1–25. [CrossRef]
22. Moustafa, N. A systemic iot-fog-cloud architecture for big-data analytics and cyber security systems: A review of fog computing. *arXiv* **2019**, arXiv:1906.01055.
23. Hsu, C.-H.; Hong, H.-J.; Elgamal, T.; Nahrstedt, K.; Venkatasubramanian, N. Multimedia fog computing: Minions in the cloud and crowd. In *Frontiers of Multimedia Research*; ACM Press: New York, NY, USA, 2017; pp. 255–286.
24. Sufyan, F.; Banerjee, A. Computation Offloading for Smart Devices in Fog-Cloud Queuing System. *IETE J. Res.* **2021**, 1–13. [CrossRef]

25. Farzai, S.; Shirvani, M.H.; Rabbani, M. Multi-objective communication-aware optimization for virtual machine placement in cloud datacenters. *Sustain. Comput. Informatics Syst.* **2020**, *28*, 100374. [CrossRef]
26. Arri, H.S.; Ramandeep, S. Energy Optimization-based Optimal Trade-off Scheme for Job Scheduling in Fog Computing. In Proceedings of the 8th International Conference on Computing for Sustainable Global Development, New Delhi, India, 17–19 March 2021.