A new strategy for reducing latency with deep learning in fog computing environment

Birane Koundoul[1], Youssou Kassé[1], Fatoumata Baldé[1], and Bamba Gueye[2]

[1] Université Alioune Diop de Bambey
{birane.koundoul,youssou.kasse, fatoumata.balde}@uadb.edu.sn
[2] Université Cheikh Anta Diop, Dakar, Senegal
bamba.gueye@ucad.edu.sn

Abstract. The data generated by connected objects is becoming increasingly numerous and often cyclical. Fog computing (FC) has emerged as an attractive solution to bring data closer to the edge, meet requirements, and manage the growing demand for data. However, network congestion produced by connected devices increases latency and energy consumption. In addition, managing similar processes in fog nodes is difficult. Some processes evolve rapidly into complicated, heterogeneous, and dynamic structures. A reduction of latency, bandwidth, and energy consumption represents issues that can be addressed by neural networks. Indeed, Deep learning can offer fast, reliable processing times on huge quantities of data. Therefore, integrating deep learning in a fog environment would be interesting. Therefore, we proposed a new strategy that enables the selection of the best fog node within a given zone by leveraging a deep learning-based LSTM model (BRFC-LSTM) and metrics such as data size, bandwidth, and the number of layers in the node.

Keywords: IoT · deep learning · system efficiency · quality of service · fog computing.

1 Introduction

In the digital world, data continues to increase while at the same time requiring minimal response time for certain applications. The Internet of Things is a paradigm of small interconnected devices such as sensors, smartphones, etc. Almost all users use these ubiquitous devices and their roles depend on the purpose and processed data type. Most of the data generated by connected objects requires efficient processing in terms of latency, confidentiality, low bandwidth, etc. which is a challenge for the cloud.

Fog computing has emerged to address these issues. The data stored and read in fog nodes are heterogeneous. According to [1], fog computing is a technology that provides users with scale (security, cognition, agility, latency, efficiency). However, connected objects cannot perform all the tasks, which is why they rely on the cloud, with its virtually unlimited computing and processing power. This makes cloud and fog computing two complementary technologies.

However, fog computing encounters certain problems [2] that deep learning attempts to improve on FC applications to provide services such as security, resource management, accuracy, delay, energy reduction, cost, data processing, and traffic modeling. This was confirmed in [3]. According to [3], fog computing technology still suffers from performance and security issues. Most of these issues are already used and managed by deep learning (DL). DL can perform fast and reliable analyses of data to discover new information for predicting and even making important decisions. According to [2], integrating DL into CF enables more in-depth analyses and more intelligent responses.

DL, an Artificial Intelligence technology, is effective for analyzing huge multimedia data such as files, images, and videos. So using DL in FC improves the quality of service. The results obtained using DL pass through numerous layers for analysis of different characteristics. In addition, the size of incoming data is reduced as it passes from one layer to another. This integration of DL into FC reduces network traffic, latency, energy consumption, etc.

However, cloud computing is essential insofar as computationally-intensive tasks need to be redirected to this environment. Communication between fog computing and the cloud enables data to be switched [4]. Tasks that require high energy consumption should be run in the cloud.

Fog computing brings data closer to end users by placing it on network devices. It aims to overcome the previous issues faced by cloud computing. With the advent of connected objects, which continue to grow significantly [5] and have given rise to big data, things are getting complicated in the fog computing environment. For resource-constrained devices such as wireless networks, set-top boxes, switches, routers, base stations, and edge devices [6], the congestion produced by these devices delays latency, increases energy consumption, and results in high bandwidth utilization [7]. This creates problems in the system as depicted by [8, 9].

In addition, the management of similar processes in fog nodes is difficult to manage in the fog environment because some processes evolve rapidly towards complicated, heterogeneous but also dynamic structures according to [10]. Consequently, integrating Deep Learning (DL) into the fog computing environment is an asset for improving applications and providing the services mentioned above. To get more in terms of service diversity and performance, fog nodes need to be improved [11].

With the use of deep learning, according to [11], the fog computing environment can benefit in many ways from DL, because it cannot solve the problems of cloud computing on its own. Applications are constantly growing and can open up new possibilities for 5G and artificial intelligence (AI) [12]. Most of these points are already used and managed by Deep Learning. Hence the integration of Deep Learning into our architecture [13] for improved quality of service.

The rest of the paper is structured as follows. Section II reviews the related work, whereas Section III describes our architecture integrating the LSTM model. Afterward, Section IV illustrates the BRFC-LSTM module for data processing. Section V presents the experimental parameters and the results obtained by making a comparison with the BRFC and DLEFN models [14]. Finally, Section VI concludes the work and gives some perspective.

2 Related Work

Deep learning is one of the solutions for data prediction. With the plethora of data and its similarities, problems are beginning to appear in the fog environment. This has led scientists and researchers to work based on the available data to predict future results. Several methods have often been used, such as simple RNN (Recurrent Neural Network), CNN (Convolutional Neural Network), LSTM (Long-Short-Term-Memory), Bi-LSTM (Bidirectional Long-Short-Term-Memory), GRU (Gated Recurrent Unit), or even traditional methods like ANN (Artificial Neural Network).

In [2], the authors predicted that in the next five years, the number of connected objects could reach 50 billion. These objects produce an enormous amount of data, some of which are similar, posing a major problem for the fog computing environment. According to Shavan et al., deep learning offers highly accurate models while reducing response time, and requires a large amount of data. This is why, according to Shavan et al., the integration of DL into the fog environment has a positive impact, as it considerably increases computing performance. The limitations of fog computing can be managed by deep learning.

In [15], the authors developed a model for the detection of weapons in surveillance videos to alert the authorities to a possible crime. They integrated deep learning into a software-defined network (SDN) architecture to support delay-sensitive applications in a fog computing environment. The SDN is also explained in this paper to take into account the constraints of multimedia traffic. After simulation, results showed that “**YOLOv5n**” is better than “**YOLOv5-lite e**” and “**YOLOv5-lite s**”. A reduction in bandwidth and system performance is noted. The mininet emulator was used to evaluate the model’s performance. This revealed an improvement of up to 75%, 14.7%, and 32.5% in terms of respectively average throughput, average jitter, and quality of service.

According to [16], fog nodes have difficulty analyzing quickly all arriving data from their applications. This can take a long time and even impact system performance. The authors address the problems of managing mobile applications that are sensitive to latency, security, and confidentiality in a wide variety of scenarios such as communication between devices, smart homes, and transport management with connected vehicles. The author has used machine learning and deep learning to analyze big data.

In this respect, the authors have reviewed the advantages of fog computing in terms of the amount of data produced by connected devices, while at the same time highlighting the limitations of FC, which DL is ready to resolve [17]. Also in the same document, the authors explain that integrating DL into the FC environment enables much deeper analysis and provides smarter mission responses. In addition, the authors in [17] demonstrate several deep learning algorithms such as MPL (Linear Probability Model), CNN (Convolutional Neural Networks), and LSTM (Long Short Term Memory).

In [14], the authors proposed a model for entrusting fog nodes with a part of deep learning (DLEFN), which they applied to the agricultural domain. DLEFN decides on the best layer, taking into account computing capacity and available bandwidth. In [14], with DLEFN, they have proposed an algorithm that selects the fog node with a sufficient number of layers to handle the request.

However, the fog node does not have the required number of layers, it takes the available number of layers to reduce the number of expected layers. Indeed, it reduces the resource capacity for the incoming task. They measure the computing capacity and bandwidth required to process the incoming data to guarantee the quality of service for transferring the result to the cloud. Afterward, the algorithm returns the maximum number of layers that the fog node can execute within its resources. According to [14], the higher the number of layers assigned to fog nodes, the lower the volumes of data transmitted to the cloud via the network, potentially reducing network congestion and the computational load on the cloud.

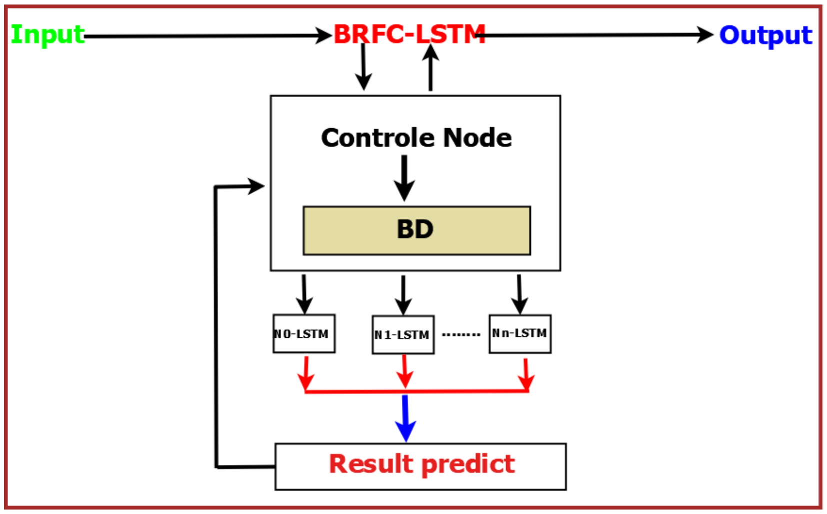

Fig. 1. Architecture of a zone by leveraging LSTM.

3 Architecture integrating the BRFC-LSTM module

This section describes a contribution that improves our former proposal [13]. Indeed, we considered a three-tier architecture composed of different layers such as IoT, fog computing, and cloud computing. At the fog computing level, we have a set of interconnected zones through a double-ring approach to facilitate data exchange between zones. Each zone has a controller node that communicates with the other nodes in the zone. Furthermore, the controller node holds some informations about each node, such as storage capacity, free memory space, processor capacity, RAM capacity, and the total number of available layers. In so doing, our algorithm can choose the most appropriate node.

With the integration of deep learning into the model, the controller node selects the best node based on the information it holds. The controller node is connected to the controller nodes of its neighbors. At the level of each node in the zone, we have integrated the LSTM model with a set of layers for data management as illustrated in Figure 1. However, if a task arrives, the controller node checks which node has more available layers for request processing. Our algorithm extracts the task by retrieving certain characteristics to determine the best fog node to assign the task. Once the node has been selected, it processes the task and returns the result. The controller then updates the node processing the request.

In addition, node selection is not based on a single criterion, namely storage capacity. Our BRFC-LSTM algorithm is based on three parameters: bandwidth, storage capacity, and the number of layers in the node. The latter is very important in neural networks, as the number of layers reduces processing time.

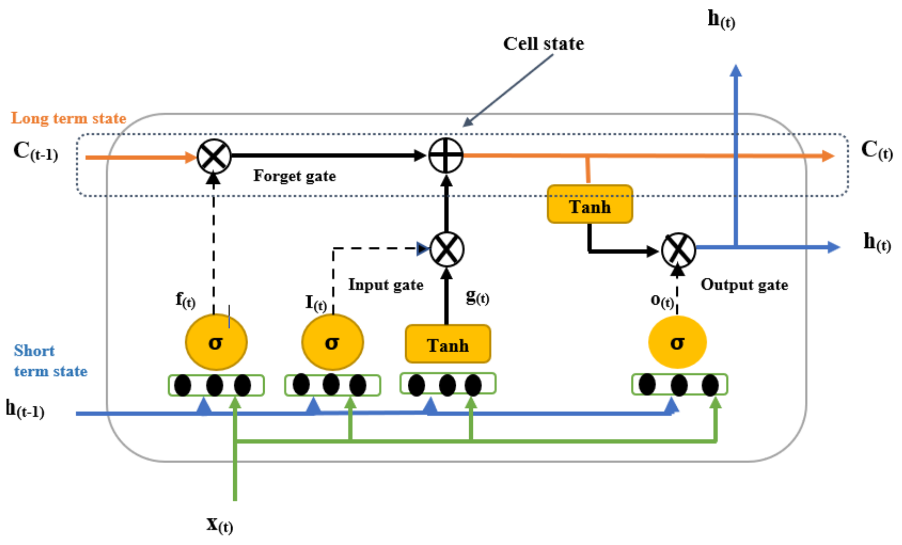

Fig. 2. The LSTM cell with its different doors.

4 Using the BRFC-LSTM with the LSTM model

An LSTM (Long Short Term Memory) is a network composed of three gates called input, forget, and output as shown in Figure 2. The input gate adds or updates new information in the network. The forget gate releases irrelevant information and finally, the exit gate transmits updated information. We have four recurrent layers with a set of units. The two short-term and long-term memories respectively represent information stored over a short and longer period.

Therefore, we have three inputs: observation data which enters the network $x_{(t)}$, the short-term memory $h_{(t-1)}$ and long-term memory $c_{(t-1)}$ which are the outputs of the previous iteration. These three inputs are managed by the three gates (inputs, forgetting, and output). Each input has an associated weight **w** which allows to trigger the activation function. Thanks to the bias $b_{(n)}$ with **n** equal to **(f, i, g and o)**, a delay of the activation function is noted. Equations 1 to 4 give the result of the inputs associated with the weights plus the bias. For instance, equations 5 to 6 give the result of the cell output.

Indeed, designed equations enable to manage the different gates and the information outputs. Equations 5 and 6 represent the outputs and, thanks to equation 5, we can keep the information much longer in the LSTM cell. This means that the information can be recovered over time.

$$f_{(t)} = \sigma(W_{xf}^T X_{(t)} + W_{hf}^T h_{(t-1)} + b_f) \tag{1}$$

$$i_{(t)} = \sigma(W_{xi}^T X_{(t)} + W_{hi}^T h_{(t-1)} + b_i) \tag{2}$$

$$g_{(t)} = \tanh(W_{xg}^T X_{(t)} + W_{hg}^T h_{(t-1)} + b_g) \tag{3}$$

$$o_{(t)} = \sigma(W_{xo}^T X_{(t)} + W_{ho}^T h_{(t-1)} + b_o) \tag{4}$$

$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \tag{5}$$

$$y_{(t)} = h_{(t)} = o_{(t)} \otimes \tanh(c_{(t-1)}) \tag{6}$$

The six equations can be divided into three groups: the gate equations, the cell equations and the final output equation. Equations 1, 2 and 4 fall into the first category.

- Equation 1 is the forgetting gate equation which specifies the information to be removed from the cell state.
- Equation 2 is the input gate, which tells us what new information we are going to store in the state of the cell. This will depend on the value returned by the sigmoid function.
- Equation 4 depicts the output gate which is used to activate the final output of the lstm block at time **t**.

For the second category of equations, we have the cell state equations.

- Equation 3 represents the candidate for the state of the cell at time(t).
- Equation 5 represents the state of the cell (memory) at time t.

And finally the third category represents the final output, which will in turn be an input to the next layer. It is represented by Equation 6.

4.1 Steps of the BRFC-LSTM algorithm

BRFC-LSTM consists of choosing the best node at the zone level. The Figure 3 describes the BRFC-LSTM algorithm for selecting the best fog node. Depending on the size of the task after extraction the controller node chooses the nodes capable of storing the task. In our architecture, we have a set of interconnected zones with fog nodes that are connected to the controller node. The controller receives all the requests and extracts the task to choose the best node. After feature extraction, our algorithm tries to find the most appropriate node for the request as shown in Figures 4 and 5. However, if the controller node verifies that

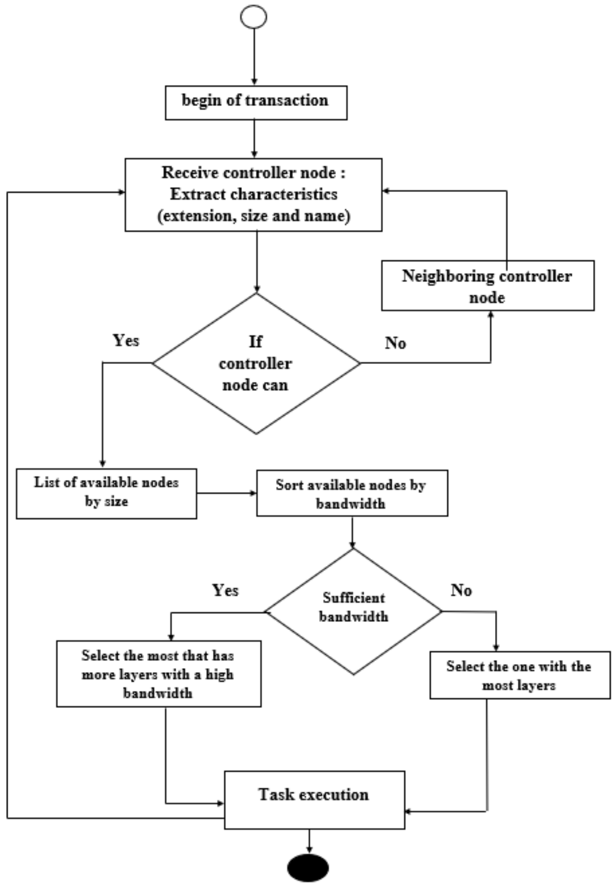
begin of transaction
Receive controller node : Extract characteristics (extension, size and name)
Neighboring controller node
Yes
If controller node can
No
List of available nodes by size
Sort available nodes by bandwidth
Yes
Sufficient bandwidth
No
Select the most that has more layers with a high bandwidth
Select the one with the most layers
Task execution

Fig. 3. BRFC-LSTM algorithm.

all the fog nodes in its zone cannot handle the request, the request is redirected to the neighboring zone. The redirection is based on the information held by the controller node. The controller node updates its tables once the request has been processed in the zone.

The sequence diagram, in Figure 4, shows the interaction between the BRFC-LSTM module and the fog nodes. This provides a list of available nodes according to size. Based on the available list, a filter is made according to the number of layers. This is a very important criterion, since in neural networks, the size of the request decreases from one layer to another. According to [14], the higher the number of layers assigned to fog nodes, the lower the volumes of data transmitted to the cloud via the network, which potentially reduces network congestion and the computing load on the cloud.

Additionally, the sequence diagram illustrated in Figure 5 shows the choice of the most appropriate node according to the number of layers about the threshold. The choice of parameters is based on empirical experiments. Three main steps summarise the choice of the best node:

- The first step is to select all the available nodes whose size is greater than the size of the request. This means that the zone receiving the request can process it. Based on the list obtained, the controller node checks the other two criteria. If the controller node cannot process the request, it is redirected to the most appropriate zone. Each zone is connected to two zones and holds information from neighboring controller nodes. This enables better redirection.
- The controller node then chooses the node with the most layers. This reduces processing time. The more layers a node has, the shorter the processing time. Priority is given to nodes with more layers than the threshold. Even if the nodes do not have more layers than the threshold, the task is still processed in the zone.
- Finally, depending on the list obtained, the nodes with more bandwidth are selected from the list. The higher the bandwidth, the lower the processing time.

5 Evaluation

In this section, we discuss the experimental parameters to obtain a result. The results obtained were compared with our BRFC model [13] and DLEFN model [14]. The BRFC model has a controller node in each zone to supervise the different nodes zone. This provides load balancing, but the placement time, execution time, and bandwidth are significant compared with the new proposal.

It is worth noticing that DLEFN algorithm does not have a global view of the system. As a result, part of a task may be handled by one node and the rest by another. In this paper, we have proposed a new architecture incorporating deep learning to improve request processing time.

Tables 1 and 2 show respectively the characteristics of a zone with all its parameters and the characteristics of a task, i.e. the resources required for a task. For our experiments, we chose the object sizes and the latency between fog nodes and between zones to represent the scenarios like [18]. Two scenarios

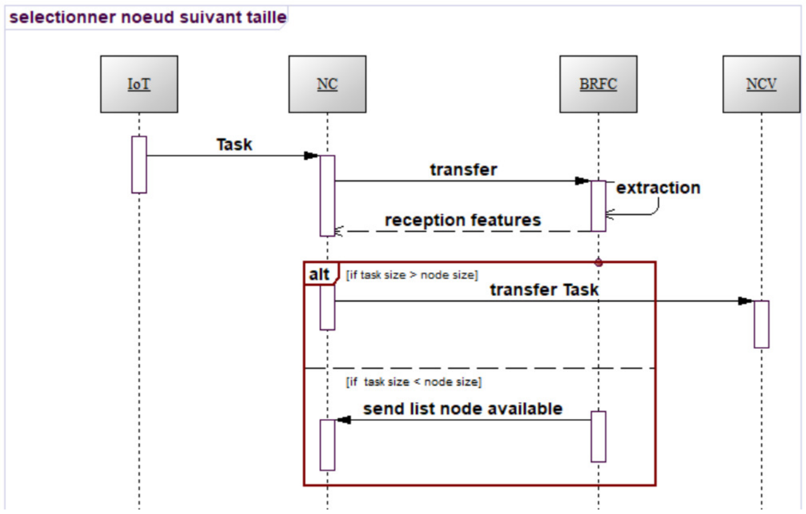

Fig. 4. Sequence diagram showing the selection of available nodes according to their sizes.

were proposed in our case: a 5-zone scenario with 20 fog nodes within each zone; a second 10-zone scenario with 10 fog nodes within each zone. Each scenario contains 100 fog nodes. The choice of the duration of each service is between 0.1 and 4.2 ms [19]; a service time of each controller node is fixed at 0.2 s [20] and the simulation duration is 1000 s like in [21].

However, in one case, the data is randomly generated and tasks arrive following the exponential distribution. In another case, we assume that the arrival of tasks follows a Poisson distribution of mean rate λ and that each controller node processes tasks with a distributed exponential service time of mean rate μ. The system can be modeled as an **M/M/1/K** queue where **K** represents the capacity of the queue and in service. We have a single server (controller node) at each zone. The total number of jobs in the controller node does not exceed **K** (the queue capacity). **FIFO** (First In First Out) is applied to the tasks in the queue. This means that the first task in the queue is served and so on. It may not be the first to complete its execution. Each experiment is run 10 times to obtain stable results.

5.1 Experimental parameters

5.2 Results

After simulation with the GridSim tool, we were able to obtain a better result compared with the BRFC models and the DLEFN model. We consider scenarios using 5 and 10 zones (sites). In our model, we generated data for five sites in our architecture with 100 nodes, i.e. 20 nodes per site. This corresponds to the

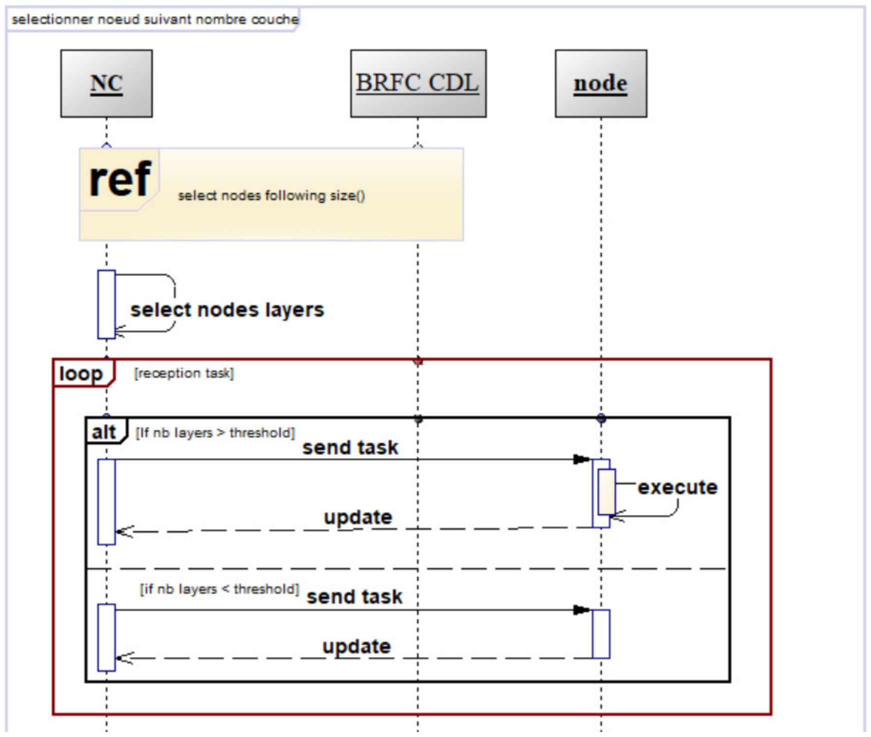
selectionner noeud suivant nombre couche
NC
BRFC CDL
node
ref
select nodes following size()
select nodes layers
loop
[reception task]
alt
[If nb layers > threshold]
send task
execute
update
[if nb layers < threshold]
send task
update

Fig. 5. Sequence diagram showing the selection of the best node according to the bandwidth and/or the number of layers.

Table 1. Characteristics of fog areas.

Parameters	**Fog**
Number of area	[5 - 10]
Number of nodes per are	[10 - 20]
Number of fog nodes in the system	100
Latency	50 - 100 ms between area
Number of tasks	24000
Bandwidth	500 - 2000 MB/S

Table 2. Characteristics of the task.

properties	**Values**
Storage capacity (MB)	[1 - 10]
Storage capacity (KB)	256
Bandwidth (MB/S)	[0,5 - 1]

first scenario. For the second scenario, we increased the number of sites and decreased the number of nodes at the zone level. The simulation was carried out on a DELL 1.8GHz Intel Core i5 8th generation dual-core machine, 8GB RAM, and 500GB SDD hard disk.

Figure 6 shows the average placement time of the random data. We used the RStudio tool with version 4.3.0 to generate the random data and the data following the exponential distribution.

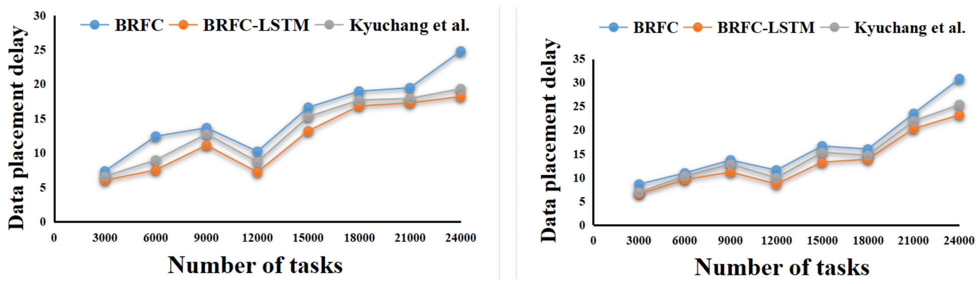

Fig. 6. Average placement time in different scenarios with random distribution.

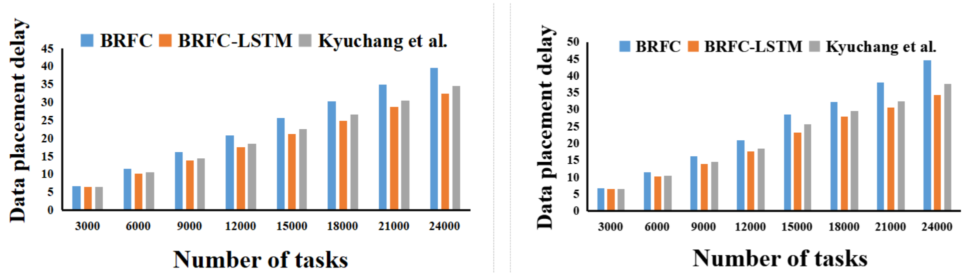

Fig. 7. Average placement time in the different scenarios with an exponential distribution.

We applied these data in our simulations for comparison with the BRFC and DFELN models. Figure 6-(a) represents the average data placement delay for 5 sites following a simulation time equal to 1000s. In Figure 6-(b), we used the same data for 10 sites. We note that our BRFC-LSTM proposal has a better data placement time than BRFC and the DLEFN proposal in [14]. We also note that the larger the number of sites, the longer the placement time. There is no great increase in terms of time, but a slight increase. In all cases, our proposal always offers a better data placement time.

In Figure 7, we used the same scenarios (5 and 10 sites) to estimate the data placement time according to an exponential distribution with a Poisson distribution, with the same remark always depending on the number of sites. We observe that the placement time increases with the number of sites. We can see that the proposed model, BRFC-LSTM, has a better placement time even when increasing the number of zones.

The placement times of the DLEFN model and ours are not very different, as both methods use deep learning. The advantage of our model over DLEFN in [14] is that the controller node in our model selects the best node based on the information it holds and has a global view of the zone and information about neighboring controller nodes.

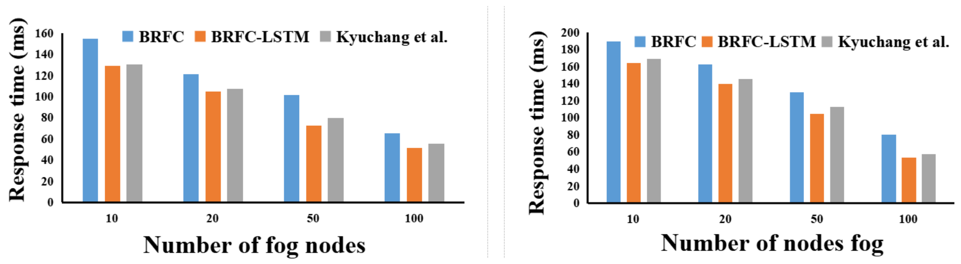

Fig. 8. Average response time for tasks in different scenarios depending on the number of fog nodes.

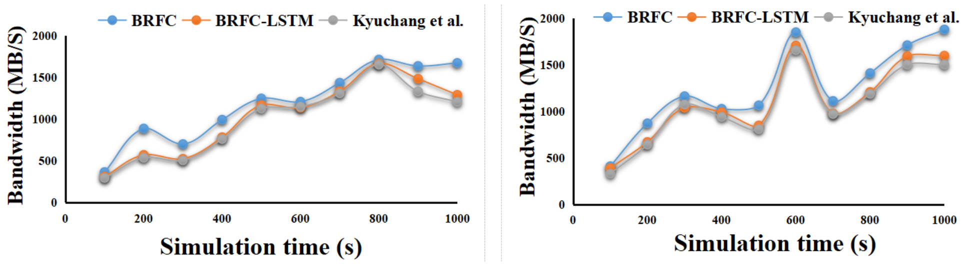

Fig. 9. Average bandwidth consumption in different scenarios with random distribution.

We also compared the average response time of the three models according to the number of fog nodes. We varied the number of fog nodes from 10 to 100 as shown in Figure 8. In Figure 8-(a), we tested with random data, and Figure 8-(b) with data following an exponential distribution. The observation is the same: as the number of fog nodes increases, response time decreases and there is less loss of non-executed data, as shown in Figure 11, where the number of lost tasks is represented. Our proposal greatly exceeds the BRFC model in terms of response time and bandwidth. This means that with the integration of deep learning in

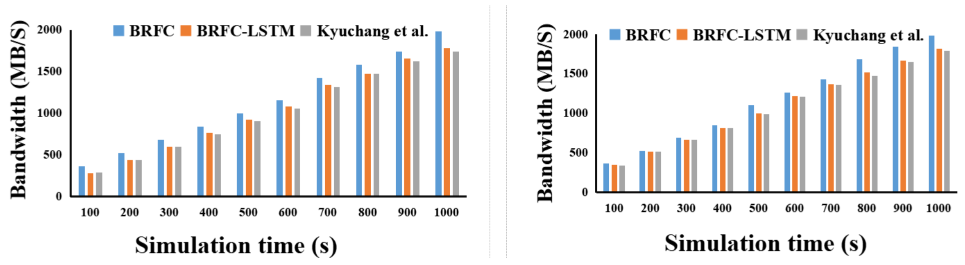

Fig. 10. Average bandwidth consumption in the different scenarios with an exponential distribution.

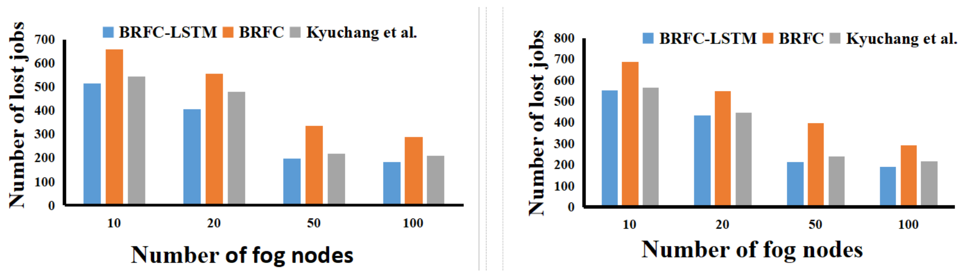

Fig. 11. Number of tasks not performed.

the fog environment, we have improved the response time, the placement delay, and also the bandwidth.

In Figure 9, we have used the bandwidth parameter. We injected 24,000 tasks at random, which arrive exponentially. Similarly, in Figure 10, we have also used the same parameter, but the data follows an exponential distribution. The bandwidth consumed increases with the simulation time. But we observe that the bandwidth consumption of our proposal is somewhat similar to that of [14]. This is in contrast to the BRFC proposal where we note the absence of deep learning. The number of layers in a fog node is very important as it reduces the query size from layer to layer. Over a period of time, the DLEFN model of [14] has better bandwidth in the 5 and 10 scenarios. The difference is not huge.

Packet losses were noted in the system. Depending on the number of fog nodes in the system, the number of lost packets decreases as shown in Figure 11. The **M/M/1/K** queue model is applied in the system with **K** the number of tasks in the queue allowed. The system cancels new jobs if the queue capacity is exceeded. This leads to the loss of the task. According to the three models, our proposal has less packet loss whatever the number of fog nodes used in the simulation.

6 Discussions

In this paper, we used deep learning to improve the response time of requests. After simulation, we obtained a better response and placement time compared to the proposal of kyuchang et al.

However, our proposal has some limitations in terms of bandwidth compared to that of kyuchang et al. Our model consumes more bandwidth because a return to the controller node is made after each data placement to perform an update. The data sent and stored in the zone nodes. The information returned by the fog node storing the data is used to update its table. The information sent by the controller node and returned by the fog node consumes bandwidth compared to the Kychang model.

This allows the controller node to easily retrieve the information during future searches.

7 Conclusion

Deep learning is an artificial intelligence technology. Deep learning models tend to work with huge amounts of data. Integrating deep learning into the fog environment improves the quality of the system.

This article describes and evaluates, through simulations, three different algorithms for placing data in a fog environment. The results show that the BRFC-LSTM model consistently achieves the best performance in almost all the simulations carried out, both in terms of execution time and data placement delay. In terms of bandwidth consumption, the DLEFN model is better than our proposed model because the round-trip at the controller node consumes significant bandwidth.

Our BFRC-LSTM algorithm, after simulation, gave a better result than the BRFC and DLEFN models. With the proposed model, we obtained a data placement delay in both scenarios and a better response time compared to the two models compared. However, we observed an increase in latency as the number of zones increased. With $5zones, theresponsetimeisshorterthanwith10$ zones. The same number of fog nodes is used for 5 and 10 zones.

In future work, we plan to develop a new module, integrating deep learning into the controller node to select a better fog node. This will allow the controller node to predict the best node. However, in terms of bandwidth, our proposal is somewhat similar to the DLEFN model. We have decided to make improvements to achieve a trade-off between response time and bandwidth. An improved model for low bandwidth consumption.

References

1. S. V. Margariti, V. V. Dimakopoulos, et G. Tsoumanis, " Modeling and Simulation Tools for Fog Computing—A Comprehensive Survey from a Cost Perspective ", Future Internet, vol. 12, no 5, Art. no 5, mai 2020, doi: 10.3390/fi12050089.
2. S. Askar, Z. Jameel, et S. Kareem, " Deep Learning and Fog Computing: A Review ", août 2021, doi: 10.5281/zenodo.5222647.
3. F. E. F. Samann, A. M. Abdulazeez, et S. Askar, " Fog Computing Based on Machine Learning: A Review ", Int. J. Interact. Mob. Technol. IJIM, vol. 15, no 12, Art. no 12, juin 2021, doi: 10.3991/ijim.v15i12.21313.
4. B. Koundoul, Y. Kasse, F. Balde, et B. Gueye, " Leveraging Cloud Inter-zone Architecture for Response Time Reduction ", in Research in Computer Science and Its Applications, Y. Faye, A. Gueye, B. Gueye, D. Diongue, E. H. M. Nguer, et M. Ba, Éd., in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Cham: Springer International Publishing, 2021, p. 87-97. doi: 10.1007/978-3-030-90556-9-8.
5. S. F. Hassan et R. Fareed, " Video streaming processing using fog computing ", in 2018 International Conference on Advanced Science and Engineering (ICOASE), oct. 2018, p. 140-144. doi: 10.1109/ICOASE.2018.8548869.
6. S. Yi, Z. Hao, Z. Qin, et Q. Li, " Fog Computing: Platform and Applications ", in 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), nov. 2015, p. 73-78. doi: 10.1109/HotWeb.2015.22.
7. I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, et S. Ullah Khan, " The rise of "big data" on cloud computing: Review and open research issues ", Inf. Syst., vol. 47, p. 98-115, janv. 2015, doi: 10.1016/j.is.2014.07.006.
8. Q. D. La, M. V. Ngo, T. Q. Dinh, T. Q. S. Quek, et H. Shin, " Enabling intelligence in fog computing to achieve energy and latency reduction ", Digit. Commun. Netw., vol. 5, no 1, p. 3-9, févr. 2019, doi: 10.1016/j.dcan.2018.10.008.
9. Z. Jameel et S. Askar, " Machine Learning Powered IoT for Smart Applications ", vol. 5, p. 92-100, févr. 2021, doi: 10.5281/zenodo.4497664.
10. K. H. Abdulkareem et al., " A Review of Fog Computing and Machine Learning: Concepts, Applications, Challenges, and Open Issues ", IEEE Access, vol. 7, p. 153123-153140, 2019, doi: 10.1109/ACCESS.2019.2947542.
11. P. K. Sharma, M.-Y. Chen, et J. H. Park, " A Software Defined Fog Node Based Distributed Blockchain Cloud Architecture for IoT ", IEEE Access, vol. 6, p. 115-124, 2018, doi: 10.1109/ACCESS.2017.2757955.
12. L. Li, K. Ota, et M. Dong, " Deep Learning for Smart Industry: Efficient Manufacture Inspection System With Fog Computing ", IEEE Trans. Ind. Inform., vol. 14, no 10, p. 4665-4673, oct. 2018, doi: 10.1109/TII.2018.2842821.
13. B. Koundoul, Y. Kasse, F. Balde, et B. Gueye, " A Dual Ring Architecture Using Controllers for Better Load Balancing in a Fog Computing Environment ", in Innovations and Interdisciplinary Solutions for Underserved Areas, in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Cham: Springer Nature Switzerland, 2022, p. 144-154. doi: 10.1007/978-3-031-23116-2-11.
14. K. Lee, B. Silva, et K. Han, " Deep Learning Entrusted to Fog Nodes (DLEFN) Based Smart Agriculture ", Appl. Sci., vol. 10, p. 1544, févr. 2020, doi: 10.3390/app10041544.
15. C. Fathy et S. N. Saleh, " Integrating Deep Learning-Based IoT and Fog Computing with Software-Defined Networking for Detecting Weapons in Video Surveillance Systems ", Sensors, vol. 22, no 14, p. 5075, juill. 2022, doi: 10.3390/s22145075.

16. P. Csr, Fog Computing, Deep Learning and Big Data Analytics-Research Directions. 2019. doi: 10.1007/978-981-13-3209-8.
17. K. D. Ahmed et S. Askar, " Deep Learning Models for Cyber Security in IoT Networks: A Review ", Int. J. Sci. Bus., vol. 5, no 3, p. 61-70, 2021.
18. B. Confais, A. Lebre, et B. Parrein, " Performance Analysis of Object Store Systems in a Fog and Edge Computing Infrastructure ", in Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXIII, A. Hameurlain, J. Küng, R. Wagner, R. Akbarinia, et E. Pacitti, Éd., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2017, p. 40-79. doi: 10.1007/978-3-662-55696-2-2.
19. X. Xu et al., " Dynamic Resource Allocation for Load Balancing in Fog Environment ", Wirel. Commun. Mob. Comput., vol. 2018, p. e6421607, avr. 2018, doi: 10.1155/2018/6421607.
20. F. Balde, H. Elbiaze, et B. Gueye, " GreenPOD: Leveraging queuing networks for reducing energy consumption in data centers ", in 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), févr. 2018, p. 1-8. doi: 10.1109/ICIN.2018.8401602.
21. B. Gueye et G. Leduc, " Resolving the Noxious Effect of Churn on Internet Coordinate Systems ", in Self-Organizing Systems, T. Spyropoulos et K. A. Hummel, Éd., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, p. 162-173. doi: 10.1007/978-3-642-10865-5-14.