IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 February 2023, accepted 13 March 2023, date of publication 15 March 2023, date of current version 21 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3257425

APPLIED RESEARCH

Neural-Hill: A Novel Algorithm for Efficient Scheduling IoT-Cloud Resource to Maintain Scalability

SANDESH ACHAR
Walmart Global Tech, Sunnyvale, CA 94086, USA
e-mail: sandeshachar26@gmail.com

ABSTRACT The age of the Fourth Industrial Revolution (4IR) is the era of smart technologies and services. The Internet of Things (IoT) is at the heart of these smart services. The IoTs are resource-constrain devices. They act as middleware in intelligent systems and maintain communications between cloud servers and smart services. Processing related to intelligent decision-making, including data processing, cleaning, feature extraction, and analysis, is performed on the cloud servers. The IoT devices respond according to the decisions the applications run on the cloud servers make. The massive number of internet-connected devices is increasing by 8% per year. The cloud infrastructure backing these enormous numbers of IoT devices must be scheduled efficiently to maintain Quality of Service (QoS). An optimized scheduling scheme is essential to minimize the cost and enhance scalability. This paper proposes an innovative and novel algorithm, Neural-Hill, which combines the Deep Neural Network (DNN) and Random Restart variant of the Hill Climbing algorithm to schedule IoT-Cloud resources efficiently and ensure scalability. It is a preemptive scheduling algorithm designed to operate in dynamic task scheduling. The performance of the Neural-Hill algorithm has been evaluated in terms of optimal solution-finding time, execution time, routing overhead, and throughput. The experimental results demonstrate the significant quality of service improvement with the assurance of better scalability.

INDEX TERMS Internet of Things, cloud computing, efficient scheduling, cloud resources, deep neural networks, hill climbing, optimization, scalability.

I. INTRODUCTION

The number of active Internet of Things (IoT) is expected to surpass 27 billion by 2025, according to the current trends [1]. A remarkable 8% growth has been observed between 2021 to 2022 when the number of active IoT devices surpassed 12 billion. The IoTs are physical devices that communicate over the internet [2]. They are connected to sensors, actuators, control units, and other devices. These devices continuously transmit data over the internet during their active periods [3]. It is no exaggeration to say that the IoTs are at the heart of modern intelligent services, including smart homes [4], smart healthcare systems [5], smart cities [2], smart parking management systems [6], and smart transportation management systems [7]. However, limited resources make IoT devices unsuitable for running Artificial Intelligence (AI) applications [8]. That is why they mainly work as middleware. These devices perceive the environment and send the data over to cloud servers. The applications running on the servers receive the data, process them, and make intelligent decisions. The network consists of IoT devices and cloud servers known as the IoT-Cloud network [9]. IoT-Cloud networks handle large volumes of traffic [10]. It is essential to schedule cloud resources optimally to process these large volumes of traffic in real-time. The novel Neural-Hill algorithm presented in this paper analyzes the traffic pattern, predicts the required cloud resources in the successive scheduling cycle, and optimally schedules the Virtual Machines (VMs) to process the IoT data in real-time.

The IoT-Cloud network uses the Infrastructure as a Service (IAAS) architecture [11]. The IoT-Cloud network remains active 24 × 7 [12]. However, there are peak hours when

The associate editor coordinating the review of this manuscript and approving it for publication was Stefano Scanzio.

the network experiences maximum traffic and off-peak hours when the network traffic is average [13]. Moreover, certain IoT networks do not exhibit any regular pattern. Cloud computing offers services on demand through the pay-as-you-go payment method [14]. An IoT cloud server becomes expensive without an efficient scheduling approach, and the QoS degrades [15]. That is why cloud resource optimization through efficient scheduling has become an active research field. There are numerous techniques to ensure it [16]. However, the IoT-Cloud has some additional characteristics apart from regular cloud computing [17]. That is why it is essential to develop methods exclusive to IoT-Cloud by focusing on unique characteristics. The Neural-Hill algorithm proposed in this paper has been developed from this observation. It schedules IoT-Cloud resources to maintain QoS and scalability efficiently.

The Neural-Hill algorithm is a combination of a Deep Neural Network (DNN) [18], and a Random Restart Hill Climbing (RRHC) algorithm [19]. The DNN predicts the VM's computation load in the successive task-scheduling cycle. Based on this prediction, the RRHC schedules the tasks for the underloaded VMs. The repetition of this process ensures optimal distribution of computational load on VMs. The following list shows the overall contributions of this research.

- Design and development of an optimized Deep Neural Network for predicting load on virtual machines running on cloud servers.
- Development and implementation of the Neural-Hill algorithm by combining DNN and RRHC.
- Performance evaluation of the novel Neural-Hill algorithm, which proves the feasibility of the proposed system to schedule IoT-Cloud resources efficiently.

The Neural-Hill algorithm is a Deep Learning technology breakthrough in IoT-Cloud resource optimization through efficient scheduling. The innovative application of predicting the state space landscape using DNN instead of creating them traditionally makes the proposed methodology unique. The rest of the article has been organized into five different sections. The literature review has been presented in the second section. The details of the methodology have been discussed in the third section. The fourth section presents the results and performance evaluation of the proposed Neural-Hill algorithm. The limitations of the paper and future scope have been highlighted in the fifth section. Finally, the paper has been concluded in the sixth section.

II. LITERATURE REVIEW

Nithiyanandam et al. [20] develop an ant colony-based algorithm to optimize IoT-Cloud resources to schedule tasks efficiently. It is a Markov chain-based approach. The combination of ant-colony and Markov chain is used to schedule resources in the IoT-Cloud environment optimally. Their methodology improves the Quality of Service (QoS). However, the ant colony-based optimization suffers from three limitations. The first limitation is the stagnation phase problem. The second limitation is the exploration and exploitation rate-dependent performance. And the third limitation is the convergence speed [21]. On the other hand, hill-climbing-based optimization is more efficient. It is faster and ideal for resource constraint systems. The proposed Neural-Hill algorithm uses the Hill Climbing (HC) algorithm for optimization. Another weakness of the methodology proposed by Nithiyanandam et al. [20] is the application of the Markov chain in IoT-Cloud resource scheduling. Markov models are inappropriate for a random time interval [22]. That means the system developed by Nithiyanandam et al. [20] is inappropriate for scheduling where the VMs are allocated for a random amount of time [20]. On the contrary, the proposed Neural-Hill algorithm uses a Deep Neural Network whose performance is not limited by random time intervals.

A scheduling algorithm Javanmardi et al. [23] developed for IoT-Fog network resource scheduling is worth attention. However, it is not directly applied to or developed for the IoT-Cloud network. It uses a Software-Defined Network (SDN) architecture. Although the algorithm seems applicable to the IoT-Cloud environment, the SDN introduces additional complexity. The proposed Neural-Hill algorithm is straightforward to integrate with the system. Another study by Apat et al. [24] addresses the increasing demand for computing resources for the Internet of Things (IoT). It aligns with one of the observations that motivate the development of the proposed Neural-Hill algorithm. However, the approach presented in [24] follows an edge computing paradigm focused on ad-hoc networks.

Sangeetha et al. [25] address the increase in demand for cloud resources from media content. They express the significance of cloud resource management to maintain the QoS. Although this methodology is not directly related to IoT-Cloud networks, applying the Gray Wolf Optimization (GWO) technique to optimize cloud resources has drawn our attention. The GWO-based framework proposed in this paper is a promising solution for optimally managing and allocating cloud resources. They have enhanced the scalability by using a Deep Neural Network (DNN) to reduce the system delay in processing and storing resources on the cloud. However, GWO-based optimization suffers from poor convergence speed [26]. It is sensitive to initialization parameters, including population size, search space, and the number of iterations. The proposed methodology uses HC-based optimization, which converges faster than GWO. At the same time, it is not sensitive to parameter initialization [27]. Another weakness of GWO is getting stuck at local minima [28]. The HC exhibits a similar weakness [29]. We used the Random Restart variation of the HC algorithm to prevent getting stuck at local minima or local maxima. The proposed Neural-Hill algorithm uses DNN similar to [25]. However, it combines the HC algorithm with it, which makes the proposed system faster, more efficient, and more robust.

A study conducted by Qu et al. [30] to optimize the QoS in cloud, fog, edge, and IoT demonstrates promising results. This research includes energy optimization as well. It encompasses a more extensive boundary where cloud, fog, edge, and IoT are formed. The IoT-Cloud context from this study aligns with the proposed Neural-Hill algorithm. This study shows that IoT-Cloud network optimization is still a vibrant field of research, and there is scope for improvements. One of the challenges they addressed is efficient resource scheduling to improve scalability. The proposed Neural-Hill algorithm is a potential solution to this limitation.

III. METHODOLOGY

The proposed Neural-Hill (NH) algorithm is a combination of a set of Deep Neural Network (DNN) blocks and the Random Restart Hill Climbing (RRHC) Algorithm. The overview of the proposed methodology has been illustrated in figure 1. The DNN blocks are constructed of a seven-layer Deep Neural Network. Each block receives the Virtual Machine (VM) processing log as input, and it predicts the load on the VM in the next cycle. Each DNN block does the same thing for different VM. The predictions from the DNN blocks form a state space landscape. The random restart hill-climbing algorithm receives the state space landscape as input. It uses the cost and objective functions to find the underloaded and overloaded VMs. The response from the random restart hill-climbing algorithm is transferred to the VM task manager. The task manager assigns new requests to the VMs, which are underloaded. This is how the proposed novel Neural-Hill algorithm optimally schedules tasks for IoT networks connected to the cloud.

1) NETWORK ARCHITECTURE

The Deep Neural Network, the DNN block's building unit, has been constructed by following a Fully Connected (FC) network architecture [31]. Following the regression principle, this network has been designed to predict a VM's operational load at a particular time [32]. It has four hidden layers with 32 hidden nodes in each layer. The experimenting network has one output node. The output from the network is governed by equation 1.

$$O^l = \sigma^l(B^l + W^l * O^{l-1}) \tag{1}$$

In equation 1, B^l is the bias at layer l. Here, W^l refers to the weight matrix of l^{th} layer. The output from every layer is generated through the convolution operation between the input signals and the layer's weight matrix. It is expressed as $W^l * O^{l-1}$ in equation 1. The signals of a layer are further processed by an activation function, which is expressed as σ in equation 1. Here, σ^l refers to the activation function of l^th layer. The network designed in this experiment uses two activation functions: Rectified Linear Unit (ReLU) [33] and Sigmoid function [34]. The ReLU, defined by equation 2, has been used for the hidden layers. The Sigmoid function is used at the output layer of the network, which is defined by equation 3.

$$\sigma_{ReLU} = max(0, x) \tag{2}$$

$$\sigma_{sigmoid} = \frac{1}{1 + e^{-x}} \tag{3}$$

The bias specified in equation 1 has been used in this research to mitigate the effect of the overfitting problem by turning on and off neurons which have been expressed in the relation presented in equation 4.

$$B = \begin{cases} 0, & off \\ 1, & on \end{cases} \tag{4}$$

A. LEARNING ALGORITHM

In this research, we used the back-propagation algorithm as the learning rule for the network [35]. We use a log-based cost function which finds the difference between the prediction and expected output from a layer. The cost function has been defined by equation 5.

$$C_f = -\frac{1}{|x|} \sum_{i=1}^{|x|} ln(P(y^i|x^i|)) \tag{5}$$

The weights of the hidden nodes are updated based on the response of the cost function of equation 5. Updating the weights optimally is essential to train the network properly. The performance of the entire Neural-Hill algorithm depends on the correct prediction from the network. We used Adaptive Moment Estimation (ADAM) optimizer to update the weights of the hidden nodes of the proposed network [36]. The ADAM optimizer used in this paper is defined by equation 6.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)[\frac{\delta C_f}{\delta \omega_t}] \tag{6}$$

The m_t in equation 6 refers to gradients that are aggregated over time t. The previous aggregated average, expressed in m_{t-1}, is multiplied by the moving average. The moving average is expressed using β. The β is a constant in this paper, which defines how much gradient will be moved on an average at the given time t. The empirical analysis shows that $\beta = 0.9$ is the optimal value of this constant. The ADAM is dependent on the derivative of the cost function, which is expressed as $\frac{\delta C_f}{\delta \omega_t}$. The ADAM aims to find the optimal values to update the weights. It has been done using equation 7.

$$\omega_{t+1} = \omega_t - \alpha m_t \tag{7}$$

The m_t in equation 7 is the m_t obtained from equation 6. It is used to modify and update the weight of the hidden nodes. The α of equation 7 is the learning rate that is dynamically updated during the network training phase.

B. DATASET PREPARATION

Different data center handles various levels of computational load. The purpose of the data center, regional location, season, special events, type of service, and many other parameters control the nature of the computational load of a data

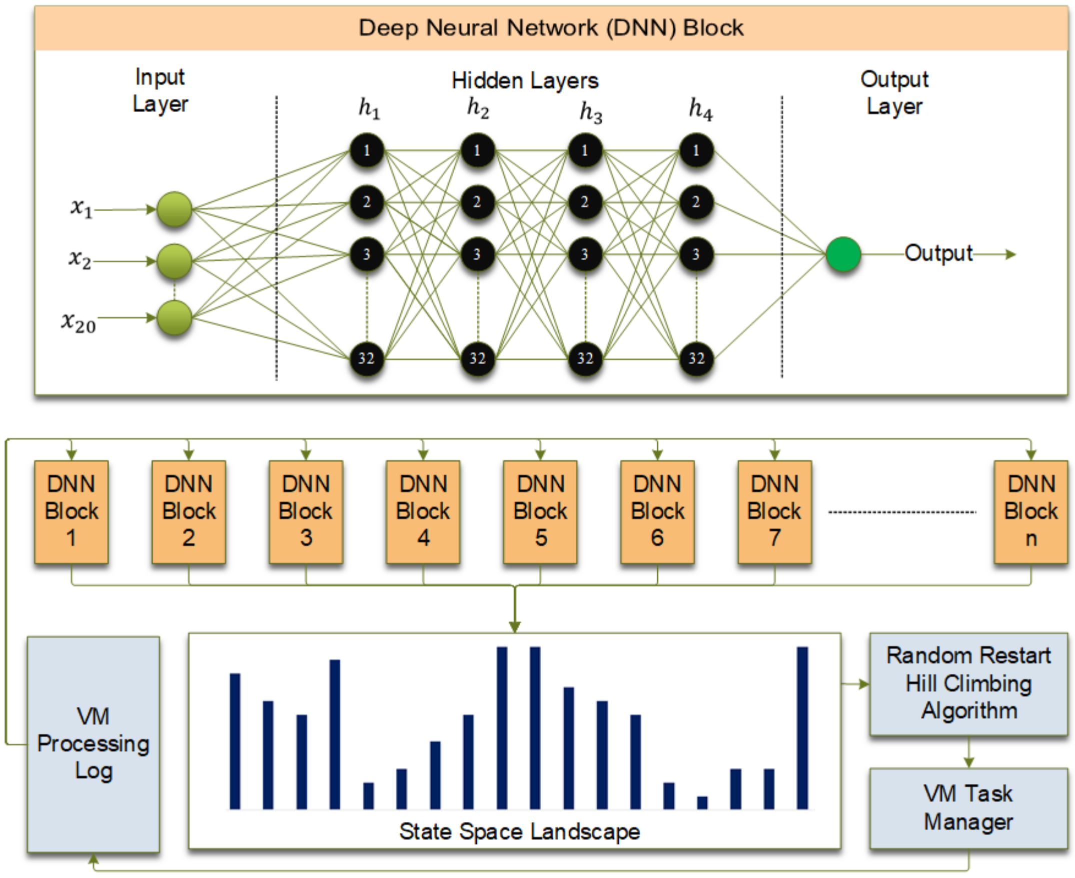

FIGURE 1. The overview of the methodology.

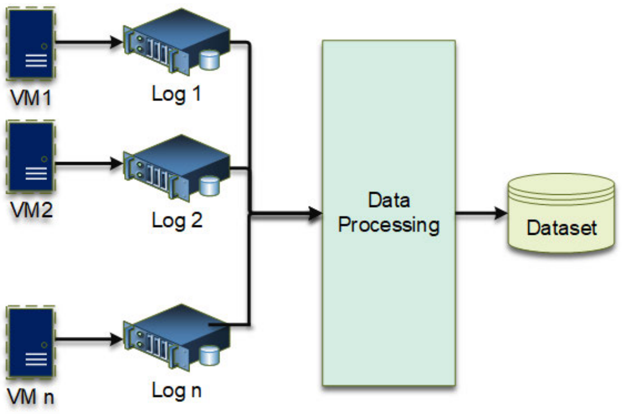

FIGURE 2. Dataset preparation.

center [37]. This is why a dataset prepared from one data center may not apply to another. This is one of the challenges of the deep learning-based cloud scheduling process. We used an innovative approach to tackle this challenge. Figure 2 illustrates our data preparation process.

Every Virtual Machine (VM) that runs on the cloud server maintains an operational log. This log includes computational load information along with other information [38]. Not every information available in the log is helpful for resource scheduling. The information related to the usage of CPU, memory, disk space, GPU, Power, and the active period of the applications is useful for scheduling cloud resources. The data related to these fields are extracted from the log during the data processing. These data are the training dataset for the proposed Neural-Hill algorithm. The proposed system uses the existing log and extracts its relevant data fields. After the extraction, it stores the data. And this is the dataset for the proposed Neural-Hill algorithm. We've already mentioned that the dataset prepared for a data center may not apply to another dataset. That is why this paper has presented the dataset preparation process instead of the dataset itself. This is an indication of the robustness of the proposed methodology.

C. TRAINING THE NETWORK

The proposed network has been trained using the dataset prepared from the process explained in the previous section. There are 17,00,000 instances in the dataset. These instances are divided into five categories which are listed in table 1. These categories are dependent on how cloud resources are used. The instances reflect the least amount of cloud resource utilization, falling into the Negligible category. In contrast,

TABLE 1. Different categories of instances of the dataset.

Serial	Type of Usage	Number of Instances	Duration of Observation
1	Negligible	321000	720 hours
2	Below Average	134000	
3	Average	731900	
4	Significant	146000	
5	Peak	367100	

instances that resulted in the highest resource use fell into the Peak Category. Below Average, Average, and Substantial categories are in between the Negligible and Choose categories.

We used a dynamic learning rate for the proposed network [39]. The initial learning rate was set to $\alpha = 0.001$, and the final learning rate is $\alpha = 0.00001$. The network learns from 10,000 epochs, with 1200 iterations in every epoch. The dataset has been divided into training, testing, and validation dataset by maintaining a ratio of $70 : 15 : 15$, respectively. The training dataset has been used to train the network, with a total of 11,90,000 instances. The testing and validation datasets contain 2,55,000 each. The training and validation datasets have been used to train and validate the system. The testing dataset was used later to evaluate the performance of the trained network. The network learns to predict the VMs' computational load with 91.31% training and 90.55% validation accuracy. The learning curve of the training process has been illustrated in figure 3.

Figure 3a shows the accuracy of the network in terms of training and validation dataset over 10,000 epochs. The 3b is about the training and validation loss. The training and validation learning curve is observable to follow a similar pattern in both accuracy and loss curves. It indicates that the network is neither overfitting nor underfitting [40]. It is observable from 3a that the accuracy increases with almost a constant slope rise. There is no sudden rise or a sudden drop in accuracy. Similar characteristics are visible in the 3b. The loss falls smoothly from 40% to 9.07%. The learning curve presented in figure 3 has not been smoothened to demonstrate the actual characteristics.

D. THE NEURAL-HILL ALGORITHM

The Neural-Hill algorithm combines the proposed Deep Neural Network (DNN) and Random Restart Hill-Climbing (RRHC) algorithm. The random restart hill climbing algorithm is a variant of the hill climbing algorithm where it is randomly restarted multiple times. In this experiment, we set the value of random restart (r) to 5. The RRHC uses the State Space Landscape (SSPL) predicted by the DNN. The proposed Neural-Hill algorithm uses objective and cost functions to determine the VMs under minimum and maximum computational load. At every iteration, the transferable processes from the highest-loaded VMs are moved to lowest loaded VMs. The proposed Neural-Hill algorithm has been presented in the algorithm 1.

The algorithm 1 uses the DNN presented in this paper to predict the SSPL. Then it finds the VMs where the maximum and minimum processes run from the predicted SSPL using

Algorithm 1 The Neural-Hill Algorithm

```
SSPL ← DNN(VM-logs)
max = obj(SSPL, 5)
min = cost(SSPL, 5)
$L_x$ = len(max)
$L_n$ = len(min)
while $L_x \neq L_n$ do
    ProcessTransfer(min ← max)
end while
function obj(SSPL, R)
current ← initial state of SSPL
while current = true do
    neighbor = highest valued neighbor of current
    if neighbor is not better than current then
        return current
    else
        current = neighbor
    end if
end while
function cost(SSPL, R)
current ← initial state of SSPL
while current = true do
    neighbor = lowest valued neighbor of current
    if neighbor is not better than current then
        return current
    else
        current = neighbor
    end if
end while
```

the RRHC algorithm at $r = 5$ from both objective and cost function perspectives. Afterward, the processes are transferred from the VM operating at the highest computational load to the lowest.

IV. RESULTS AND PERFORMANCE EVALUATION

The proposed Neural-Hill algorithm combines the Random Restart Hill Climbing (RRHC) algorithm and Deep Neural Network (DNN). The DNN is the predictor, and RRHC is the optimizer. The predictor predicts the operational load on VMs, and the optimizer optimizes the load by allocating processes to underloaded VMs. As a result, the entire system stays ahead of time and optimally schedules the resources before the IoT network requests those resources. This section presents the details of experimental results and performance evaluation of the proposed system. First, the performance of the DNN is presented. After that, the experimental setup is discussed. Finally, the performance of the Neural-Hill algorithm in cloud resource scheduling for IoT networks has been discussed.

A. PERFORMANCE OF THE DNN

The performance of the Neural-Hill algorithm-based efficient cloud resource scheduling to optimize IoT-cloud performance and enhance the scalability depends on the accurate

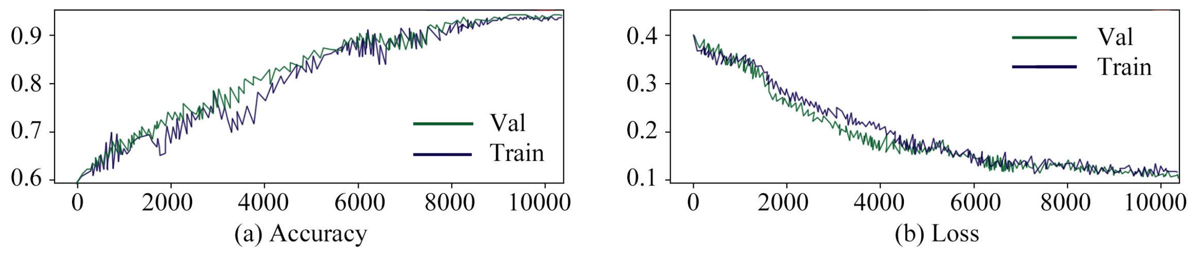

FIGURE 3. Learning curve.

prediction of the DNN. We used state-of-the-art evaluation metrics to evaluate the performance of the DNN. The first evaluation metric is the Coefficient of Determination (R^2). It indicates how well the experimenting DNN predicts the outcomes [41]. The second evaluation metric is the Root Mean Square Error (RMSE). It measures the average difference between the predicted and actual computational loads. The third evaluation metric is the Mean of Absolute Error (MAE). It shows the significance of an error the DNN may make while predicting the SSPL [42]. The fourth and last metric is the Mean Absolute Percentage Error (MAPE). It is the sum of absolute errors in the prediction divided by the ground truth [43]. The R^2, RMSE, MAE, and MAPE are defined in the equation 8, 9, 10, 11, respectively.

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(a_i - pi)^2}{\sum_{i=1}^{m}(a_i - mean(a))^2} \tag{8}$$

$$RMSE = \sqrt{\frac{1}{m} \times \sum_{i=1}^{m}(p_i - a_i)^2} \tag{9}$$

$$MAE = \frac{1}{m} \times \sum_{i=1}^{m}|p_i - a_i| \tag{10}$$

$$MAPE = \frac{1}{m}\sum_{i=1}^{m}|\frac{p_i - a_i}{a_i}| \tag{11}$$

Here in equation 8, 9, 10, and 11, the a_i is the ground truth in the dataset. The p_i is the target variable the network predicts, and m is the number of instances on the dataset.

B. EXPERIMENTAL SETUP

The effectiveness of the Neural-Hill algorithm was investigated in an experimental environment. The experimental setup includes a data center and physical and virtual IoT devices. Replicating an IoT-Cloud network with physical nodes is expensive. That is why We used both physical and virtual IoT devices.

1) DATA CENTERS

The proposed methodology has been implemented in an experimental setup to analyze and evaluate the performance. This experiment used a 64-bit Ubuntu Server Operating System (OS). The experimental setup comprises two data centers, 10 Virtual Machines (VMs), and 150 to 2000 simulated dummy tasks. The experimenting data centers use Dell PowerEdge R940 Rack Server computers with 8 Solid State Hard Drives (SSDs) for each server. It is powered by a 2× Intel®Xeon®Gold 6252 processor. This processor has 24 cores with 48 threads. The maximum clock speed is 10.4 GT/s. It has 35.75MB Cache memory. This device has 32GB dual rank of primary memory distributed in 4 slots. These primary memories have 3200 MT/s data transmission capability. These server computers have used the PERC H330 Adapter FH international storage controller. It has a total storage capacity of 15.36TB connected to eight different SSD ports. These storage devices have a maximum 6Gbps data transmission capability.

2) IoT DEVICES

We used Raspberry Pi 4 Model B as the IoT device. It has 4GB primary memory with a storage capacity of 32 GB. It is powered by a Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit processor at 1.5GHz clock speed. Because of the resource constraints, we used two physical Raspberry Pi 4 Model B and 18 simulated versions of the same device. Each IoT device has 8 sensors connected to it. There are a total of 160 active sensors in the experimental settings.

C. EVALUATION METRICS

We used four different evaluation metrics to analyze the performance of the proposed scheduling algorithm. These are the Optimal Solution Finding Time (OSFT), Execution Time (ET), Routing Overhead (RO), and Throughput (TP).

1) OPTIMAL SOLUTION FINDING TIME (OSFT)

IoT devices are resource-constrained systems. That is why they are not suitable for heavy computing. In most cases, they perform basic computations and forward the data to the cloud server. The IoT-cloud servers are responsible for receiving the data from the IoT devices and locating the appropriate application intended to process the IoT data. The time the IoT-cloud server takes to search for optimal solutions for a particular IoT node is the Optimized Solution Finding Time (OSFT).

2) EXECUTION TIME (ET)

In order to choose the best virtual machines (VMs) for an IoT node, the proposed framework uses a set of criteria.

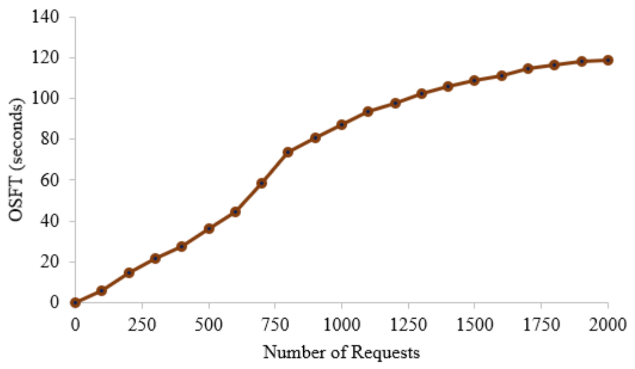

FIGURE 4. **Optimal Solution Finding Time (OSFT).**

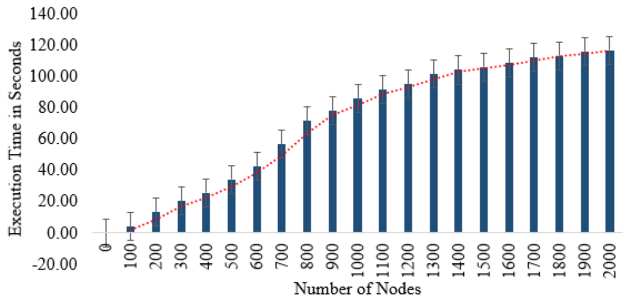

FIGURE 5. **The execution time with respect to the number of IoT nodes.**

Usually, there are some time delays when doing so. The term ''execution time'' is used to describe this lag. The proposed framework demonstrates excellent performance in optimally allocating resources. However, an execution time delay is a sensitive issue in IoT-cloud servers because IoT devices don't have extensive computational and storage capabilities. At the same time, they transmit and receive responses from different connected systems serving real-time services, making it essential for the IoT-cloud servers to respond in real-time. Suppose the framework optimally allocates resources within a short period. In other words, it takes longer execution time. In that case, the overall service quality of the system will be unacceptable.

3) ROUTING OVERHEAD (RO)

The routing overhead is the quantity of the packets sent from the IoT devices for maintenance and to keep track of the route for discovery. Each IoT device is connected to the IoT cloud server. This communication happens through packet switching circuits without any dedicated route. The IoT devices generate small packets to update the routing table frequently. For a single device, the routing overhead is ignoble. However, an IoT-cloud server serves thousands of IoT nodes. As a result, routing overhead becomes a vital performance measurement parameter for IoT-cloud servers.

4) THROUGHPUT

IoT nodes are diversely distributed throughout the network. Most of these nodes don't have high bandwidth because of being resource-constrained systems. Even if the intermediate devices have high bandwidth, IoT devices' limitations impose challenges on the overall data exchange rate. This is where the throughput becomes an important performance evaluation criterion.

D. PERFORMANCE ANALYSIS

1) OSFT ANALYSIS

We analyzed the performance of the proposed framework using the evaluation metrics discussed in the previous subsection. The first evaluation criterion is the Optimal Solution Finding Time (OSFT). We evaluated the OSFT with a maximum of 2,000 user requests simultaneously. We divided the range from 0 to 2000 requests equally with a 250-step gap to analyze the performance easily. The performance of the experimenting framework is illustrated in figure 4. The analysis demonstrates the scalability of the proposed framework. The time to discover the optimal path and find the solution for 0 to 600 requests increases slowly. However, there is a sharp rise from 600 to 750 requests. After that, the slope of the curve falls further. The resource allocation becomes more stable for 750 to 2000 requests. It proves that the proposed Neural-Hill algorithm optimally allocates the resources and maintains scalability even if the number of requests increases.

2) ET ANALYSIS

The proposed framework has been designed to simultaneously process requests from multiple IoT devices. Each request is optimally assigned to an intended application running on some certain VM. Request distribution, processing, and response transmission time for different numbers of nodes have been illustrated in figure 5. The ET analysis shows similar characteristics to the OSFT analysis. The time to find the optimal solution and time to execute the request is expected to follow a similar pattern which has been achieved through the proposed framework. This means the experimenting framework optimally distributes the resources and makes the IoT-cloud server scalable.

3) RO ANALYSIS

The Routing Overhead (RO) analysis illustrated in figure 6 demonstrates that the proposed framework reduces the routing overhead. Figure 6 shows both routing overhead before and after using the Neural-Hill algorithm. It shows noticeable improvement made by the proposed algorithm over the performance of the existing algorithms.

The effect of the Neural-Hill algorithm on RO has been further studied with a spider chart illustrated in 7. The RO is directly related to the period the applications run on the cloud server and respond to the IoT requests. It is very impressive that the proposed algorithm does not significantly impact RO when the active period of the applications is less than 15 seconds. Usually, RO increases simultaneously with the

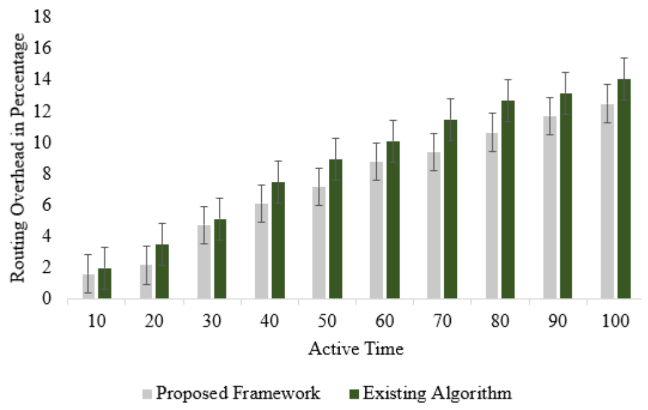

FIGURE 6. The routing overhead of the proposed framework and existing solution.

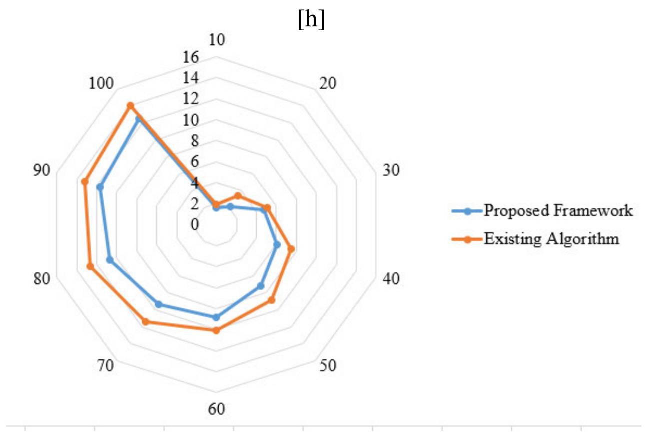

FIGURE 7. The routing overhead analysis.

increment of the active period of the algorithm. However, it shows different characteristics when the Neural-Hill algorithm is applied. This algorithm reduces the rate of increment of routing overhead. This behavior is observable for longer active periods only. That means the proposed algorithm enhances the scalability of the IoT-Cloud network.

The overhead routing reduction in percentage has been listed in table 2. The percentage improvement is marginally visible in figure 6 and 6 because of the scaling factor. However, it is easily noticeable in table 2. The maximum overhead reduction is 22.26% for 40 seconds of active time, and the minimum overhead reduction is 8.02% for 30 seconds of active time.

E. DYNAMIC SCHEDULING

The experimenting Neural-Hill algorithm dynamically schedules the tasks in every Task Cycle (T). It ensures optimal task distribution to every virtual machine. After optimal distribution, it dynamically relocates the tasks from right most VM to other VMs to optimize resource usage. Figure 8 illustrates the experimental results up to the fifth task cycle. The initial approach of this research was to schedule the tasks based on continuous task influx dynamically. However, it initiates temporary service disruption during the allocation and relocation of the tasks. The revised approach maintains discrete task cycles where $\{T|T \in N, 1 \leq T \leq \infty\}$ and applies the algorithm in every task cycle. As a result, the ongoing processes are not disrupted. The tasks on the active VMs are relocated in between the task cycle when the existing responses are sent to the IoT network, and subsequent requests are assigned to the VMs. As a result, process disruption becomes insignificant. Figure 8 demonstrate the dynamic scheduling of the tasks to different VMs at different task cycle. It is noticeable that the VM10 becomes idle after the fifth task cycle.

TABLE 2. Routing overhead reduction in percentage.

Active Time	Proposed Framework	Existing Algorithm	Percentage Reduction
10	1.62	1.98	22.22
20	2.17	3.47	59.91
30	4.74	5.12	8.02
40	6.11	7.47	22.26
50	7.16	8.91	24.44
60	8.78	10.11	15.15
70	9.4	11.47	22.02
80	10.65	12.65	18.78
90	11.7	13.17	12.56
100	12.48	14.04	12.50

V. LIMITATION AND FUTURE SCOPE

The Neural-Hill algorithm performs remarkably efficiently, scheduling IoT-Cloud resources and improving scalability. However, it is not immune to limitations. There are several limitations of this algorithm which have been discussed in this section. However, instead of limitations, these have been considered as future scope to conduct more experiments and further strengthen the algorithm.

A. SCHEDULING SCHEME

There are different types of scheduling, for example, immediate scheduling, batch scheduling, static scheduling, dynamic scheduling, preemptive scheduling, etc. The proposed Neural-Hill algorithm works with preemptive scheduling only. However, there are scopes of modifying the algorithm to apply it to other scheduling schemes. It paves the path to further research and enables the Neural-Hill algorithm to cover various scheduling schemes.

B. EXPERIMENT IN LABORATORY SETUP

The proposed Neural-Hill algorithm has been experimented with in an experimental laboratory environment. Although a real-world cloud server has been used, the researcher prepared the IoT infrastructure used in this experiment. The real-world scenario may deviate from the experimental setup. This is one of the limitations of this paper. An initiative has been taken to apply the Neural-Hill algorithm in a medium-scale private IoT-Cloud network to analyze its impact of it in a real-world scenario. The findings will be presented in subsequent research.

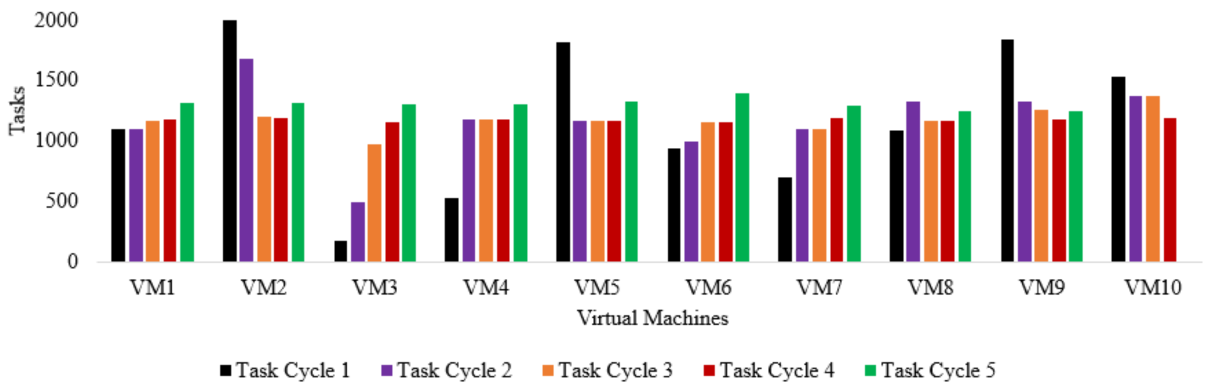

FIGURE 8. **A subset of the dynamic scheduling up to the fifth task cycle.**

C. DATASET LIMITATION

The dataset prepared for the Neural-Hill algorithm applies to similar cloud servers to the experimenting server. However, it is unlikely that different cloud servers will have similar usage patterns. That means there is no assurance that the dataset prepared for one IoT-Cloud network can be effectively used for another network. It is a general limitation of the IoT-Cloud research domain and an active field of research for future endeavors.

The limitations of the proposed Neural-Hill algorithm do not undermine its potential of it. They reflect the opportunity to conduct more experiments to develop better versions.

VI. CONCLUSION

The IoT-Cloud is more dynamic than traditional cloud computing. The rapid growth of IoT devices has made IoT-Cloud resource optimization a challenging task and appealing research field. The Neural-Hill algorithm has been developed to optimally and efficiently schedule IoT-Cloud resources to maintain scalability. IoT-Cloud networks handle large volumes of traffic and massive process requests. The volume of the request varies depending on multiple factors. As a result, optimally allocating cloud resources is a major concern. This is where the Neural-Hill algorithm demonstrates the contributions. It takes advantage of Deep Learning technology to predict the computational load. As a result, the system knows the number of service requests about to come beforehand. The state space landscape for the hill climbing algorithm is produced through this prediction. Applying the random restart variants of the hill climbing algorithm discovers the underloaded and overloaded virtual machines. It ensures that all available VMs handle an almost equal amount of processes at a particular time, leaving room for more requests. As a result, the IoT-Cloud network experience maximum scalability. The unique concept of combining Deep Neural Network and Random Restart Hill Climbing algorithm presented in this paper maintains scalability even at 2000 simultaneous requests. It minimizes the execution time by distributing tasks to idle VMs. A remarkable 22.26% routing overhead reduction has been achieved through this innovative algorithm as well. The Neural-Hill algorithm's unique architecture, efficient design, and impressive performance signify its impact on the IoT-Cloud resource scheduling research domain. However, the effectiveness of the Neural-Hill algorithm has been studied only for dynamic task scheduling in a laboratory environment. The dataset prepared and used in this research is exclusive to the experimenting cloud server. These limitations of the Neural-Hill algorithm pave the path to the future scope of conducting subsequent experiments. The methodology of this paper will be further developed to encompass dynamic and static scheduling. There is another scope for expanding the scope of the dataset to generalize it so that it can be used for cloud servers operating in different settings and serving different purposes. This future scope will be explored, implemented, and studied in subsequent versions of the Neural-Hill Algorithm.

REFERENCES

[1] R. A. Abdelouahid, O. Debauche, and A. Marzak, "Internet of Things: A new interoperable IoT platform. Application to a smart building," *Proc. Comput. Sci.*, vol. 191, pp. 511–517, Jan. 2021.

[2] A. Rejeb, K. Rejeb, S. Simske, H. Treiblmaier, and S. Zailani, "The big picture on the Internet of Things and the smart city: A review of what we know and what we need to know," *Internet Things*, vol. 19, Aug. 2022, Art. no. 100565.

[3] E. Fleisch, "What is the Internet of Things? An economic perspective," *Econ., Manag., Financial Markets*, vol. 5, no. 2, pp. 125–157, 2010.

[4] H. Sequeiros, T. Oliveira, and M. A. Thomas, "The impact of IoT smart home services on psychological well-being," *Inf. Syst. Frontiers*, vol. 24, no. 3, pp. 1009–1026, Jun. 2022.

[5] N. Faruqui, M. A. Yousuf, M. Whaiduzzaman, A. K. M. Azad, A. Barros, and M. A. Moni, "LungNet: A hybrid deep-CNN model for lung cancer diagnosis using CT and wearable sensor-based medical IoT data," *Comput. Biol. Med.*, vol. 139, Dec. 2021, Art. no. 104961.

[6] A. Floris, S. Porcu, L. Atzori, and R. Girau, "A social IoT-based platform for the deployment of a smart parking solution," *Comput. Netw.*, vol. 205, Mar. 2022, Art. no. 108756.

[7] X. Yang, G. Liu, Q. Guo, H. Wen, R. Huang, X. Meng, J. Duan, and Q. Tang, "Triboelectric sensor array for Internet of Things based smart traffic monitoring and management system," *Nano Energy*, vol. 92, Feb. 2022, Art. no. 106757.

[8] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022.

[9] T. Pflanzner and A. Kertesz, "A taxonomy and survey of IoT cloud applications," *EAI Endorsed Trans. Internet Things*, vol. 3, no. 12, Apr. 2018, Art. no. 154391.

[10] F. Metzger, T. Hoßfeld, A. Bauer, S. Kounev, and P. E. Heegaard, "Modeling of aggregated IoT traffic and its application to an IoT cloud," *Proc. IEEE*, vol. 107, no. 4, pp. 679–694, Apr. 2019.

[11] S. Hammoudi, Z. Aliouat, and S. Harous, "A new infrastructure as a service for IoT-cloud," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2018, pp. 786–792.

[12] J. Betty Jane and E. Ganesh, "Big data and Internet of Things for smart data analytics using machine learning techniques," in *Proc. Int. Conf. Comput. Netw., Big Data IoT*. Cham, Switzerland: Springer, 2020 pp. 213–223.

[13] Z. Chen, C. B. Sivaparthipan, and B. Muthu, "IoT based smart and intelligent smart city energy optimization," *Sustain. Energy Technol. Assessments*, vol. 49, Feb. 2022, Art. no. 101724.

[14] W. Voorsluys, J. Broberg, and R. Buyya, "Introduction to cloud computing," *Cloud Comput., Princ. Paradigms*, pp. 1–44, Feb. 2011.

[15] X. Zheng, L. D. Xu, and S. Chai, "QoS recommendation in cloud services," *IEEE Access*, vol. 5, pp. 5171–5177, 2017.

[16] P. S. Rawat and P. Gupta, "Application of optimization techniques in cloud resource management," *Bio-Inspired Optim. Fog Edge Comput. Environments, Princ., Algorithms, Syst.*, pp. 73–90, Jan. 2023.

[17] P. P. Ray, "A survey of IoT cloud platforms," *Futur. Comput. Informat. J.*, vol. 1, nos. 1–2, pp. 35–46, 2016.

[18] D. Bau, J.-Y. Zhu, H. Strobelt, A. Lapedriza, B. Zhou, and A. Torralba, "Understanding the role of individual units in a deep neural network," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 48, pp. 30071–30078, Dec. 2020.

[19] E. Bytyçi, E. Rogova, and E. Beqa, "Combination of genetic and random restart hill climbing algorithms for vehicle routing problem," in *Proc. Int. Conf. Artif. Intell. Appl. Math. Eng.* Cham, Switzerland: Springer, 2020, pp. 601–612.

[20] N. Nithiyanandam, M. Rajesh, R. Sitharthan, D. S. Sundar, K. Vengatesan, and K. Madurakavi, "Optimization of performance and scalability measures across cloud based IoT applications with efficient scheduling approach," *Int. J. Wireless Inf. Netw.*, vol. 29, no. 4, pp. 442–453, Dec. 2022.

[21] J. Yu, X. You, and S. Liu, "A heterogeneous guided ant colony algorithm based on space explosion and long–short memory," *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 107991.

[22] T. DelSole, "A fundamental limitation of Markov models," *J. Atmos. Sci.*, vol. 57, no. 13, pp. 2158–2168, Jul. 2000.

[23] S. Javanmardi, M. Shojafar, R. Mohammadi, V. Persico, and A. Pescapè, "S-FoS: A secure workflow scheduling approach for performance optimization in SDN-based IoT-fog networks," *J. Inf. Secur. Appl.*, vol. 72, Feb. 2023, Art. no. 103404.

[24] H. K. Apat, B. Sahoo, S. Mohanty, and K. S. Sahoo, "A scalable software-defined edge computing model for sustainable smart city Internet of Things (IoT) application," in *Hemant Kumar Apat, Bibhudatta Sahoo, Sagarika Mohanty, Kshira Sagar Sahoo*. Boca Raton, FL, USA: Chapman & Hall/CRC, 2022, pp. 125–148.

[25] S. B. Sangeetha, R. Sabitha, B. Dhiyanesh, G. Kiruthiga, N. Yuvaraj, and R. A. Raja, "Resource management framework using deep neural networks in multi-cloud environment," in *Operationalizing Multi-Cloud Environments*. Cham, Switzerland: Springer, 2022, pp. 89–104.

[26] H. Xie, L. Zhang, and C. P. Lim, "Evolving CNN-LSTM models for time series prediction using enhanced grey wolf optimizer," *IEEE Access*, vol. 8, pp. 161519–161541, 2020.

[27] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, and A. Jafarian, "Improved chaotic binary grey wolf optimization algorithm for workflow scheduling in green cloud computing," *Evol. Intell.*, vol. 14, no. 4, pp. 1997–2025, Dec. 2021.

[28] M. Banaie-Dezfouli, M. H. Nadimi-Shahraki, and Z. Beheshti, "R-GWO: Representative-based grey wolf optimizer for solving engineering problems," *Appl. Soft Comput.*, vol. 106, Jul. 2021, Art. no. 107328.

[29] A. Kishor and P. K. Singh, "Empirical study of grey wolf optimizer," in *Proc. 5th Int. Conf. Soft Comput. Problem Solving, SocProS*, vol. 1. Cham, Switzerland: Springer, 2016, pp. 1037–1049.

[30] Z. Qu, Y. Wang, L. Sun, D. Peng, and Z. Li, "Study QoS optimization and energy saving techniques in cloud, fog, edge, and IoT," *Complexity*, vol. 2020, pp. 1–16, Mar. 2020.

[31] J. Zhao, F. Deng, Y. Cai, and J. Chen, "Long short-term memory–fully connected (LSTM-FC) neural network for $PM_{2.5}$ concentration prediction," *Chemosphere*, vol. 220, pp. 486–492, Apr. 2019.

[32] J. Kossaifi, Z. C. Lipton, A. Kolbeinsson, A. Khanna, T. Furlanello, and A. Anandkumar, "Tensor regression networks," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 4862–4882, Jan. 2020.

[33] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.

[34] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *Proc. Int. Workshop Artif. Neural Netw.* Cham, Switzerland: Springer, 1995, pp. 195–201.

[35] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," *Nature Rev. Neurosci.*, vol. 21, no. 6, pp. 335–346, 2020.

[36] Y. Arouri and M. Sayyafzadeh, "Adaptive moment estimation framework for well placement optimization," in *ECMOR XVII*. European Association of Geoscientists & Engineers, 2020, pp. 1–15.

[37] Y.-P. Xu, R.-H. Liu, L.-Y. Tang, H. Wu, and C. She, "Risk-averse multi-objective optimization of multi-energy microgrids integrated with power-to-hydrogen technology, electric vehicles and data center under a hybrid robust-stochastic technique," *Sustain. Cities Soc.*, vol. 79, Apr. 2022, Art. no. 103699.

[38] K. Perumal, S. Mohan, J. Frnda, and P. B. Divakarachari, "Dynamic resource provisioning and secured file sharing using virtualization in cloud azure," *J. Cloud Comput.*, vol. 11, no. 1, pp. 1–12, Sep. 2022.

[39] X. Wu, Y. Zhang, M. Shi, P. Li, R. Li, and N. N. Xiong, "An adaptive federated learning scheme with differential privacy preserving," *Future Gener. Comput. Syst.*, vol. 127, pp. 362–372, Feb. 2022.

[40] H. Zhang, L. Zhang, and Y. Jiang, "Overfitting and underfitting analysis for deep learning based end-to-end communication systems," in *Proc. 11th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2019, pp. 1–6.

[41] L. J. Saunders, R. A. Russell, and D. P. Crabb, "The coefficient of determination: What determines a useful R^2 statistic?" *Investigative Ophthalmol. Vis. Sci.*, vol. 53, no. 11, pp. 6830–6832, 2012.

[42] T. Chai and R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)," *Geosci. Model Develop. Discuss.*, vol. 7, no. 1, pp. 1525–1534, 2014.

[43] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, Jun. 2016.

SANDESH ACHAR was the Director of cloud engineering, leading the Site Reliability Engineering Team, the Program Management Team, and the Database Engineering Team, spread globally, for the fastest-growing, multi-million-dollar product line with Workday Inc. He is currently a Senior Manager of Engineering with Walmart Global Tech. He is a Cloud Distributed Computing Architect and an Engineering Leader with experience building, scaling, and managing a "world-class" engineering team for leading Fortune 1000 organizations. In his experience with Intuit Inc., he has successfully led the migration of more than a hundred enterprise systems to a multi-cloud environment. His most recent scholarly articles have been published in international engineering journals on forensics, greener cloud, cloud security, artificial intelligence, machine learning, and observability.